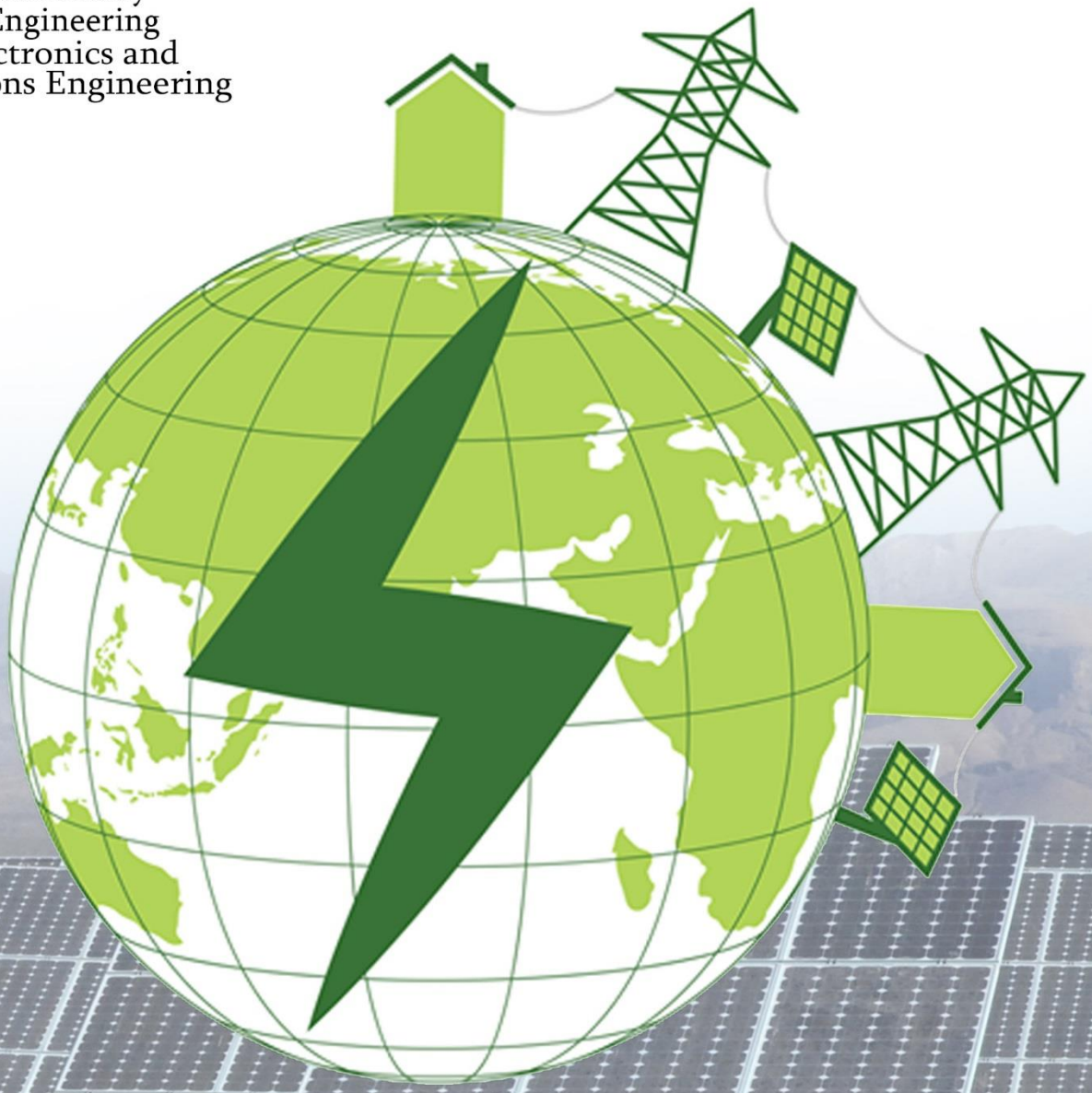
Mansoura University
Faculty of Engineering
Dept. of Electronics and
Communications Engineering



Implementation of DC Micro Smart Grid using Renewable Energy

supervisor
Prof: Sherif Kishk

PROJECT TEAM MEMBERS

Noha Abdullateef Abo Al-khair Moustafa

Yassmin Mahmoud Mahmoud Dief

Kareem Mousaad Ebraheem Ragab

Muhammed Sedeeq Antar

Ahmad Alaa Eldeen El Akabawy

Muhammed Talaat Ali Zidan

Amr Elsaied Muhammed El-Desoky

Acknowledgment

Firstly, we thank Allah for helping us in this project

We would like to thank our project supervisor

Dr. Sherif Kishk

For his efforts and his great support which helped us to improve our performance and knowledge through his helpful discussions, generous support and constructive suggestions while preparing this project.

We also thank the staff members and colleagues in the department of electronics and communications engineering faculty of engineering, Mansoura University who supported us in many ways throughout all stages of this project.

Last but not least, Special thanks for our parents for their endless efforts and unconditional support in all phases of our life especially in our education which lead us to this far of education.

Project team..

Abstract

With a renewable source of energy (solar cells) for generating electricity we will manage the distribution and storing of energy according to the needs and demands of customers with respect to their loads, priority and money. By using solar cells, the generating dc power will be equally distributed to the customers(homes), the user will be able to record the information about his loads like: power consumption, duration, number, efficiency of usage, ...etc. And either there is any extra power to their needs or lack in power, the system will be able to manage the demand and storage in an efficient and smart way.

Generating electricity from a renewable source is a good solution to reduce pollution and is a guarantee to the continuity of electricity. Also this project will help in some way to reduce population density in Delta and serves investments in different places in Egypt.

Managing electricity according to the needs and priorities of customers using smart, compatible and effective way of pricing and storage will help in taking decisions either automatically or remotely by using mobile phone, this process presents an innovative solution to solve the problem.

Contents

Chapter (4): Raspberry Pi.

Chapter (5): Server.

Chapter (6): Android.

Chapter (1): Introduction

Chapter 1
Introduction

1.1 The Problem:

The Electricity production is a major problem in Egypt while Construction of power plants requires a huge budget and take a long time. "DC Smart Micro Grid" is a small scale system which uses Solar cells to produce electricity. It is a Smart system that depends on IOT to produce, manage and utilize electricity that will serve remote areas where no electricity infrastructure. So it is a fast, permanent, economic and clean solution for generating electricity.
The claims: Daily power outage in cities due to limited number of power plants that serve a large number of users, also difficulties in serving new cities with electricity, The high cost being spent every year to solve this problem, Egypt has contracted with Siemens to 8 billion euros for the construction of 3 power plants –as mentioned by the government- and this large sum for a country suffers economically, Average per capita electricity in Egypt is 1782 kW, which is less than the world average of 2730 kW.

1.2 Why Micro grids?

Micro grid demonstrations and deployments are expanding around the world. Although goals are specific to each site, these micro grids have demonstrated the ability to provide higher reliability and higher power quality than utility power systems and improved energy utilization. The vast majority of these micro grids are based on AC power transfer because this has been the traditionally dominant power delivery scheme. Independently, manufacturers, power system designers and researchers are demonstrating and deploying DC power distribution systems for applications where the end-use loads are natively DC, e.g., computers, solid-state lighting, and building networks. These early DC applications may provide higher efficiency, added flexibility, reduced capital costs over their AC counterparts. Further, when onsite renewable generation, electric vehicles and storage systems are present, DC-based micro grids may offer additional benefits. Early successes from these efforts raises a question—can a combination of micro grid concepts and DC distribution

systems provide added benefits beyond what has been achieved individually?

1.3 DC Micro grids?

DC power systems are becoming commonplace and ubiquitous, such as in building communications and IT networks, building automation and fire life safety and security systems, on- site renewable power generation, onsite storage, remote homes, vehicles, vessels, aircraft, and powering remote communications devices. The basis for AC as the sole platform is eroding and reevaluation is timely. Compared to AC power, the distribution of DC power over DC networks has potential to provide several benefits to equipment manufacturers, electricity customers, electrical systems, and the environment including but not limited to:

1-higher power system efficiency because of fewer AC-to-DC, DC-to-AC, or AC-to-AC power conversions in local power systems that include a significant amount of distributed generation or storage that naturally produce DC power

2-higher reliability in those same systems because fewer power conversions require fewer power electronic components, with fewer potential points of failure

3-lower capital cost because of fewer power electronic components and potential reductions in conductor cost because DC allows higher current carrying capability

4-a potential for lower control system complexity and higher survivability when subject to external and internal disturbances because of the elimina-.tion of synchronization requirements of AC systems

5-higher power quality and disturbance survivability because of the power electronics and (potentially) storage buffer between the DC .microgrid and the AC grid

6-"Managed DC" technologies have communications integrated with power enabling control and configuration capabilities not present with today’s AC technology.

1.4 Metrics For Assessing the Impacts of DC Networks and Microgrids:

Assessing the total impacts (both positive and negative) of a DC microgrid beyond a traditional AC system or AC microgrid requires a common set of metrics:

1-Safety and Protection—The ability of the power system to prevent injury to people and protect equipment from damage

2-Reliability—AC power system reliability is typically measured using metrics such as System Average Interruption Duration Index (SAIDI) and System Average Interruption Frequency Index (SAIFI) that measure the customer-averaged outage duration and interruption frequency, respectively. This study reinterprets these metrics with a focus on the reliability of power supplied to individual loads or load classes within micro grids.

3-Capital Costs—Total equipment and installation costs for the microgrid as incurred by the owner of the microgrid. The distribution utility that serves a DC microgrid may also see capital cost benefits. However, it is not clear how the microgrid owner could monetize these benefits, and these benefits to the greater system are not considered here.

4-Energy Efficiency—Measured at the system level, the total input electricity required to serve an end use function. For the ownership model considered

5-Operating Costs—Present value of total variable cost (primarily energy use, but also including ancillary revenue streams, maintenance, etc) for a power system to serve an end use function.

6-Engineering Costs—Site specific engineering costs to integrate the compo- nents of the microgrid with each other and the microgrid to the surrounding power systems.

7-Environmental Impact—The total CO_2 emissions produced by marginal electricity generator used to deliver the net electrical needs at the interface of the microgrid and the local power system.

8-Power Quality—The ability of a AC or DC microgrid to stay within its respective Computer Business Equipment Manufactures Association

(CBEMA)/Information Technology Industry Council (ITIC) curve for a given type of disturbance.

9-Resilience—Ability to serve electrical load when the main AC grid is unavailable for extended periods of time, e.g. for typical multi-day outage times following major natural disasters such as hurricanes or earthquakes

1.5 DC Micro grid Applications:

Several types of AC and DC microgrids applications were compared. The initial comparison is done using several generic microgrid architectures to reveal the importance of and to initially screen the different metrics. Then, several specific microgrid applications are considered to draw out possible unique advantages of DC over AC microgrids. These applications include:

1-DC Microgrid as an Efficient, Low-Cost Platform for Economic (Steady- State).

2-High Survivability DC Micro grids.

3-Low Power Network with Differentiated and Automatically Evolvable Power Quality and Reliability.

4-Converting AC Systems to Hybrid AC/DC Systems.

5-Mobile and Remote Applications.

6-Data Center Support Systems.

7-Coupling a DC Micro grid to a HVDC Line.

8-Electric Vehicles for Backup/Emergency Power.

1.6 Project Description:

Our project is divided mainly into these Blocks(they will be explained in details in the next chapters) :
1) Power system: Which is responsible for generating, storing DC power. Consists of solar panels(10 watt power source), 18V -boost buck converter, Voltage regulator 5V and Rechargeable batteries each of 12V.

2) Power meter: Measures Generated power from solar panels, Measures transmitted power to neighbors, Measures received power from neighbors.

3) Control system: Controls on loads by 18f877A using RS-485(to provide long distances), Using Raspberry pi to control/manage homes communicating by I2C protocol. monitoring on the output power from solar cells and dissipated power in loads, managing demand and pricing of power when it is needed.

4) Mobile app: Show all information will be required for user (bill, charge, etc.), On/off loads, show power trading offers, some advice about power planning and trading benefits.
5) Server: Is used to analysis data that receives from raspberry pi(data about loads that comes from home units), tells users about conditions of their loads, suggests recommenditions, and gives permission to user to put the price of Electricity.

Chapter (2): Power Measurement Techniques

Chapter 2
Power Measurement Techniques

2.1 Introduction:

In this chapter,the concept of electric power is first introduced, and the most popular methods and instruments in dc.
Power is defined as the work performed per unit time. So,dimensionally, it is expressed as joules per second J s^{-1} .
According to this general definition,electric power is the electric work or energy dissipated per unit time and,dimensionally. It yields:

$$Js^{-1} = JC^{-1} \times Cs^{-1} = V \times A$$

Where J = joules
S = Seconds
C = Coulombs
V = Volts
A = Ampers

The product voltage times current gives an electrical quantity equivalent to power.

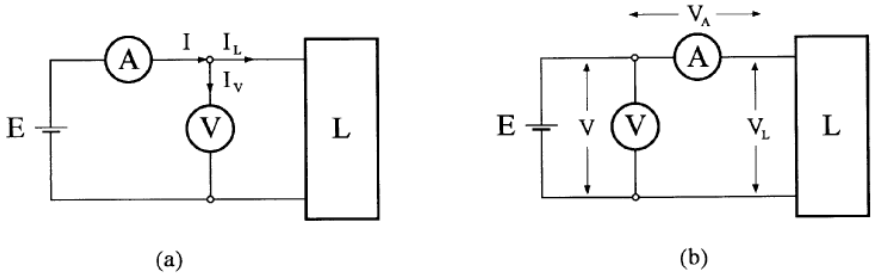
2.2 Power measurement in DC Circuits

Electric power (**P**) Dissipated by a load (**L)** fed by a dc power supply (**E**) is the product of the voltage across the load ($\mathbf{V_L}$) and the current flowing in it (***I***):

$$P = V_L \times I_L$$

Therefore,a power measurement in a dc circuit can be generally carried out using a voltmeter(V) and an ammeter (A) according to one of arrangements shown in Figure 1. In the arrangements of Figure 1(a), the ammeter measures the current flowing into the voltmeter,as well as that into the load; whereas in the arrangement of figure 1(b), this error is avoided, but the voltmeter measures the voltage drop across the ammeter

in the addition to that dropping across the load. Thus, both arrangements give a surplus of power measurements absorbed by the instruments. The corresponding measurements errors are generally referred to as insertion errors.



Two arrangement of dc power measurement circuit

According to the notation:

- I current measured by the ammeter. (Figure (a))
- V Voltage measured by the voltmeter. (Figure (b))
- R_V,R_A internal resistance of the voltmeter and ammeter.
- R_L Load Resistance
- I_v current flowing into the voltmeter.(Figure (a))
- V_A Voltage drop across the ammeter.(Figure (b))

The following expressions between the measured power and electrical power P and the measured power **V * I** are derived by analyzing the circuits of figures (a) and figure (b) respectively:

$$P = V_L * I_L = V * I * (R_V - R_L/R_V) \qquad (2.3)$$

$$P = V_L * I_L = V * I * (R_L - R_A/R_L) \qquad (2.4)$$

If:

- I_v compared with I

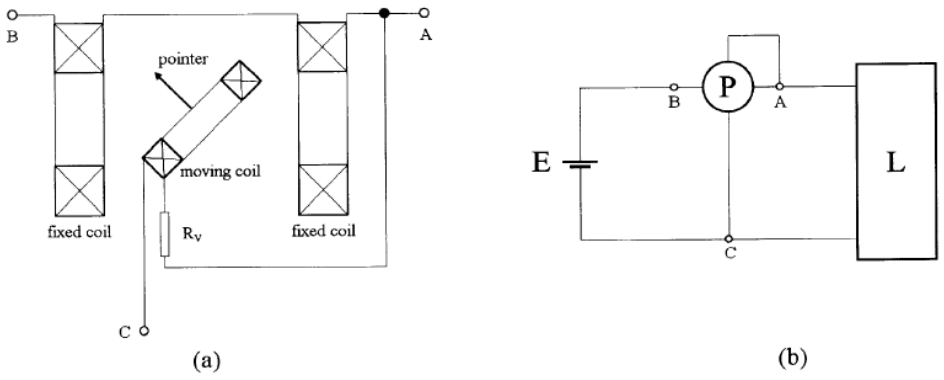
- V_A compared with V

are neglected for the arrangements of figure 1(a) and figure 1(b), respectively, it approximates yields :

$$I_V / I = R_L/R_V+R_L=R_L/R_V=0; \quad V_A / V = R_A /R_A+R_L=R_A /R_L = 0; \qquad (2.5)$$

Consequently, measured and measurand power will be coincide.
On the basis, from equations (2.3),(2.4) and (2.5), analytical corrections of the insertion errors can be easily derived for the arrangement of figures of (a) and (b), respectively.
The instrument most used in power measurements is the **dynamometer**. It is build by two fixed coils, connected in series and positioned coaxially with space between them ,and a moving coil, placed between the fixed coils and equipped with a pointer
The torque produced in the dynamometer is proportional to the product ofcurrent flowing into the fixed coils times that in moving coil. The fixed coils,generally referred to as current coils, carry the load current while moving the coil, generally referred to as voltage coil ,carries a current that is proportional,via the multiplier resistor R_V ,to the voltage across the load resistor.

R_L as a consequence, the deflection of a moving coil is proportional to the power dissipated into the load.



Power measurement with a dynamometer. (a) working principle; (b)measurement circuit

As for the case of figure (a), insertion errors are also present in the dynamometer power measurement. In particular, by connecting the voltage coil between A and C (Figure (b)), the current coils carry the surplus current flowing into the voltage coil. consequently , the power P_L dissipated in the load can be obtained by the dynamometer reading P as:
$\boldsymbol{P_L = P - V^2/R'_V}$

Where R'_V is the resistance of the voltage circuit ($R'_V = R_V + R_{VC}$) where R_{VC} is the resistance of the voltage coil. By connecting the moving coil between B and C, this current error can be avoided, but now the voltage coil measures the surplus voltage across the current coil. In this case, the corrected value is:

$\boldsymbol{P_L = P - I^2 R_c}$
Where R_c is the resistance of current coil.

2.3 Why do we measure signal levels?

A component or system's output signal level is often the critical factor in the design and ultimately the purchase and performance of almost all RF and microwave equipment. Measurement of the signal level is critical at every system level, from the overall system performance to the fundamental devices. The large number of signal measurements and their importance to system performance dictates that the measurement equipment and techniques be accurate, repeatable, traceable, and convenient. In a system, each component in a signal chain must receive the proper signal level from the previous component and pass the proper signal level on to the succeeding component. If the output signal level becomes too low, the signal becomes obscured in noise. If the signal level becomes too high, though, the performance goes nonlinear and distortion results. Or worse!

2.4 why measure power ?

The first question is why measure power at all, rather than voltage? While it is true that very accurate and traceable voltage measurements can be performed at DC, this becomes more difficult with AC. At audio and low

RF frequencies (below about 10 MHz), it can be practical to individually measure the current and voltage of a signal. As frequency increases, this becomes more difficult, and a power measurement is a simpler and more accurate method of measuring a signal's amplitude.
As RF signals approach microwave frequencies, the propagation wavelength in conductors becomes much smaller, and signal reflections, standing waves, and impedance mismatch can all become very significant error sources. A properly designed power detector can minimize these effects and allow accurate, repeatable amplitude measurements. For these reasons, POWER has been adopted as the primary amplitude measurement quantity of any RF or microwave signal.

There are many reasons it may be necessary to measure RF power. The most common needs are for proof-of-design, regulatory, safety, system efficiency, and component protection purposes, but there are thousands of unique applications for which RF power measurement is required or helpful.

In the communication and wireless industries, there are usually a number of regulatory specifications that must be met by any transmitting device, and maximum transmitted power is almost always near the top of the list. The FCC and other regulatory agencies responsible for wireless transmissions place strict limits on how much power may be radiated in specific bands to ensure that devices do not cause unacceptable interference to others. Although the real need is usually to limit the actual radiated energy, the more common and practical regulatory requirement is to specify the maximum power which may be delivered to the transmitting antenna.
In addition to the regulatory issues, transmitter power needs to be limited in many communication systems to allow optimum use of wireless spectral and geographical space. If two transmitting devices are operating in the same frequency band and physical proximity, receivers can have a more difficult time discriminating the signals if one signal is much too large relative to the other. Even in commercial broadcast, the transmitting power of each broadcast site is licensed and must be constantly monitored to ensure that operators do not interfere with other stations occupying the same or nearby frequencies in neighboring cities.

Controlling transmission power is particularly necessary in modern cellular networks, where operators constantly strive to maximize system capacity and throughput. Many modern wireless protocols use some form of multiplexing, in which multiple mobile transmitters (for example, cellular handsets) must simultaneously transmit data to a common base station. In these situations, it is necessary to carefully monitor and control the transmitted power of each device so that their signals arrive at the base station with approximately equal amplitudes. If one device on a channel has too much power, it will "step on" the transmission of other devices sharing that channel, and make it impossible for the base station to decode those signals.

Another power control issue in cellular systems is due to the close proximity of base stations in congested areas. If a device is transmitting with too much power, it will not only interfere with signals in its own cell, but can possibly interfere with the transmissions of devices in neighboring cells. Mobile devices for these systems typically implement both open-loop and closed-loop, real-time power control of their transmitters. Without accurate power control of every single device within range of a base station, cellular network capacity can be severely degraded.

Too much power has other dangers as well. For higher power systems, too much RF power can present biological hazards to personnel and animals. Safety limits are often placed on transmitted power to protect users and bystanders from the dangers of high-power RF radiation. A good example of the potential dangers of RF energy is a common microwave oven, which can severely burn human flesh just as easily as it can heat a meal. Radio and RADAR transmitters operate at still higher power levels, and present their own special hazards. It is hypothesized that even low-power RF transmitting devices such as cell phones may have potential to cause lasting biological effects. In all of these cases, there will be times when the actual power present must be monitored to ensure compliance with safety standards or guidelines.

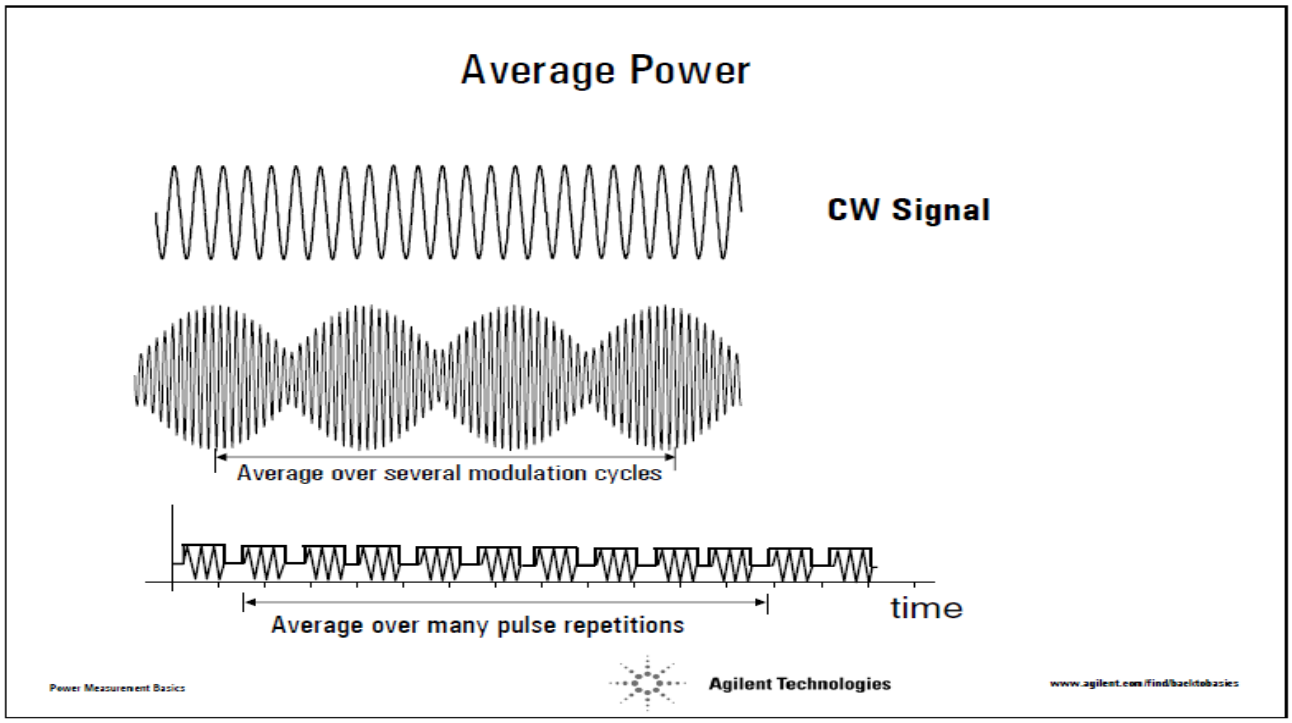
Measuring power is important for circuit designers as well. Any electronic device can be overloaded or damaged by too high a signal. Too much steady-state power can cause heating effects and destroy passive and active components alike. Too much instantaneous (“peak”) power can overstress semiconductor devices, or cause dielectric breakdown or arcing in passive components, connectors, and cables. But even at power levels well below the damage threshold of the circuit components, excessive power can cause overload of system, clipping, distortion, data loss or a number of other adverse effects. Similarly, insufficient power can cause a signal to fall below the noise floor of a transmission system, again resulting in signal degradation or loss.

2.5 Power measurements types:

There are three different types of power measurement, Average, peak, and time gated power measurements. Average power provides average power delivered over several cycles and typically is implied when talking about "power". Peak power is the maximum instantaneous power and is required on many of today’s complex wireless modulation systems. Finally, Time Gated power measurements allow both peak and average measurements to be made in the time domain, which is of particular interest to TDMA systems such as GSM.
Although a variety of instruments measure power, the most accurate instrument is a power meter and a sensor. The sensor is an RF power-to-voltage transducer. The power meter displays the detected voltage as a value of power in log (dBm) or linear (watts) units. Typical power meter instrumentation accuracy will be in the order of hundredths of a dB, while other instruments (i.e., spectrum analyzers, network analyzers) will have power measurement accuracies in the tenths of dBs or more. We will discuss the overall uncertainty of power measurements in more detail later on in this presentation. One of the main differences between the instruments is that of frequency selective measurements. Frequency selective measurements attempt to determine the power within a specified bandwidth. The traditional Power Meter is not frequency selective in that it measures the average power over the full frequency range of the sensor and will include the power of the carrier as well as any harmonics which may be generated. A Spectrum Analyzer provides a frequency selective

measurement since it measures in a particular Resolution Bandwidth. The lack of frequency selectivity is the main reason that Power Meters measure down to around -70 dBm and instruments such as a spectrum analyzer can measure much lower than this if narrow resolution bandwidths are used.

Average Power provides the average power delivered over several cycles and this is the most common power measurement performed today. Average power is defined as the energy transfer rate averaged over many periods of the lowest frequency in the signal. Average power is also defined as the power averaged over a specified time interval.



$$p_{avg}(t) = \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} p(t-\tau)$$

For an AM signal, averaging is taken over many modulation cycles, and for a pulse modulated signal the signal is averaged over several pulse repetitions. Of all the power measurements, average power is the most frequently measured because convenient measurement equipment with highly accurate and traceable specifications is available. Additional

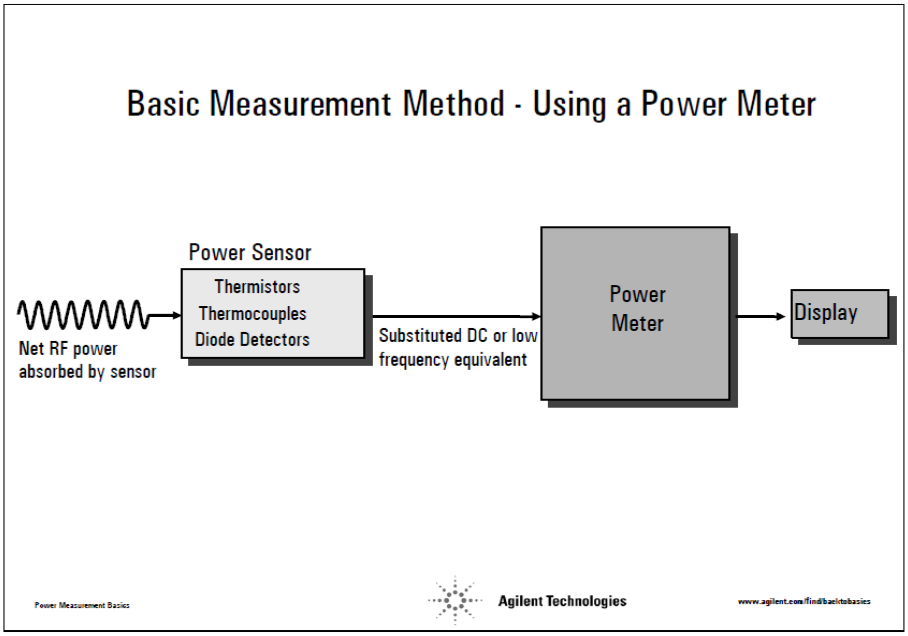
waveform information can sometimes be calculated from average power measurements if certain waveform characteristics are known. If, for example, the duty cycle of a rectangular pulsed signal is known, then peak power can be found from the average power measurement by the equation:

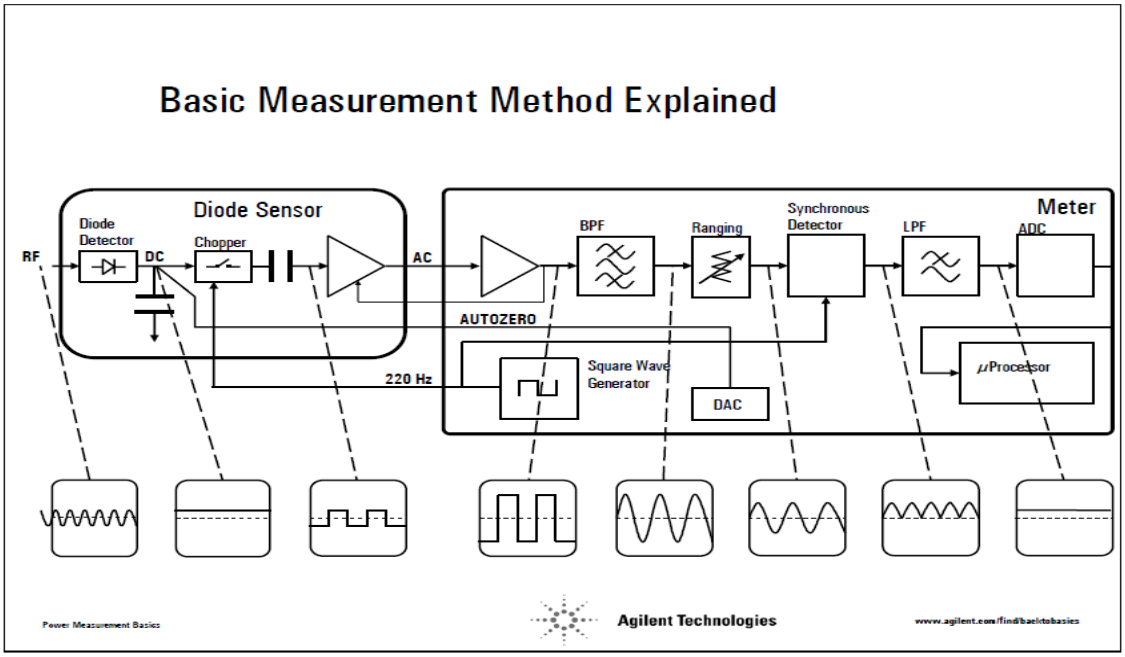
$$P_{peak} = \frac{P_{avg}}{DutyCycle}$$

Note that the peak power here is only applicable for a true rectangular pulse with no overshoot present.

Let's now look at the types of hardware, both sensors and meters, that are used in average power
measurements.

The basic idea behind a power sensor is to convert high frequency power to a DC or low frequency signal that the power meter can then measure and relate to a certain RF power level. The three main types of sensors are thermistors, thermocouples, and diode detectors. There are benefits and limitations associated with each type of sensor. We will briefly go into the theory of each type and then talk about the advantages and limitations associated with each sensor.



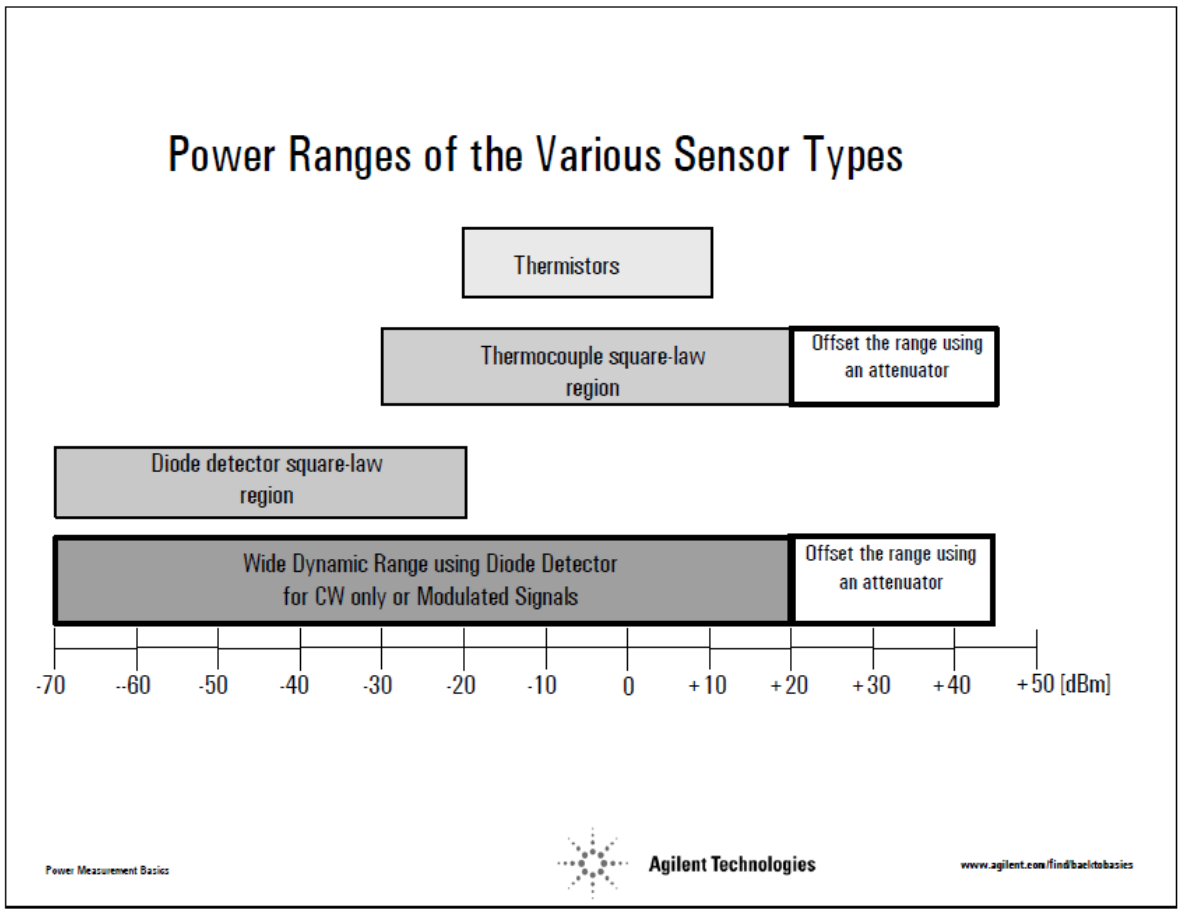


Shown here is the basic power measurements method. Both thermocouple and diode detector mounts generate voltages on the order of 100 nV. Such small voltages require choppers, AC amplifiers, and synchronous detectors to accurately detect. Power measurements with both kinds of sensors need a power-reference oscillator with a precisely known power output to adjust the calibration of the power meter to fit the particular sensor being used.

The DC output from either the thermocouple or the diode detector is very low-level (on the order of nV or V),so it is difficult to transmit on an ordinary cable because small, undesired thermocouple effects affect the measurement. For this reason Agilent includes the low-level DC circuitry in the power sensor, so only relatively high-level signals appear on the cable. To handle such low DC voltage, you must "chop" the signal to form a square wave, amplify this with an AC-coupled system, then synchronously detect the high-level AC. The chopper and first AC amplifier are included in the power sensor itself.

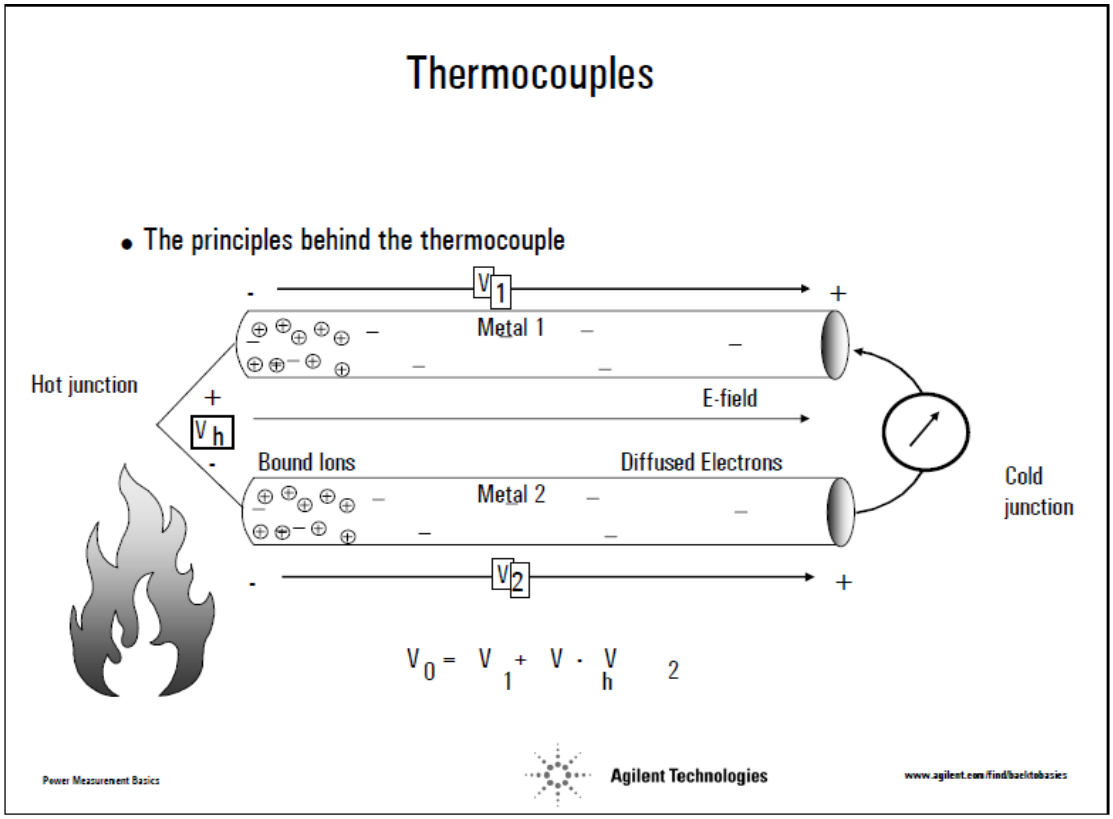
The DC output from either the thermocouple or the diode detector is very low-level (on the order of nV or V),so it is difficult to transmit on an ordinary cable because small, undesired thermocouple effects affect the measurement. For this reason Agilent includes the low-level DC circuitry in the power sensor, so only relatively high-level signals appear on the

cable. To handle such low DC voltage, you must "chop" the signal to form a square wave, amplify this with an AC-coupled system, then synchronously detect the high-level AC. The chopper and first AC amplifier are included in the power sensor itself.



Thermistors offer high accuracy, but have a more limited operating range than a thermocouple or diode detector sensor. Thermistor mount specifications are for the range from -20 dBm to +10 dBm.
Thermocouples cover a very large range of powers. Their true square-law region is from -30 dBm to +20dBm, and with an attenuator can operate up to +44 dBm. Three families of thermocouple sensors cover the complete -30 to +44 dBm range. The A- Series covers -30 to +20 dBm, the H-Series covers from -10 to+35 dBm, and the B-Series covers from 0 to +44 dBm.
Diode detectors (D-Series) have the best sensitivity, allowing them to work well below -20 dBm (stated range is -70 to -20 dBm), but above -20 dBm they begin to deviate substantially from the square-law detection region.

The wide dynamic range power sensors are diode sensors and can provide up to 90dB dynamic range. They either work by correcting for the deviation (CW Power Sensors) or by using the two path technique to allow modulated measurements. Wide dynamic range measurements can be made up to a maximum power of +44dBm.

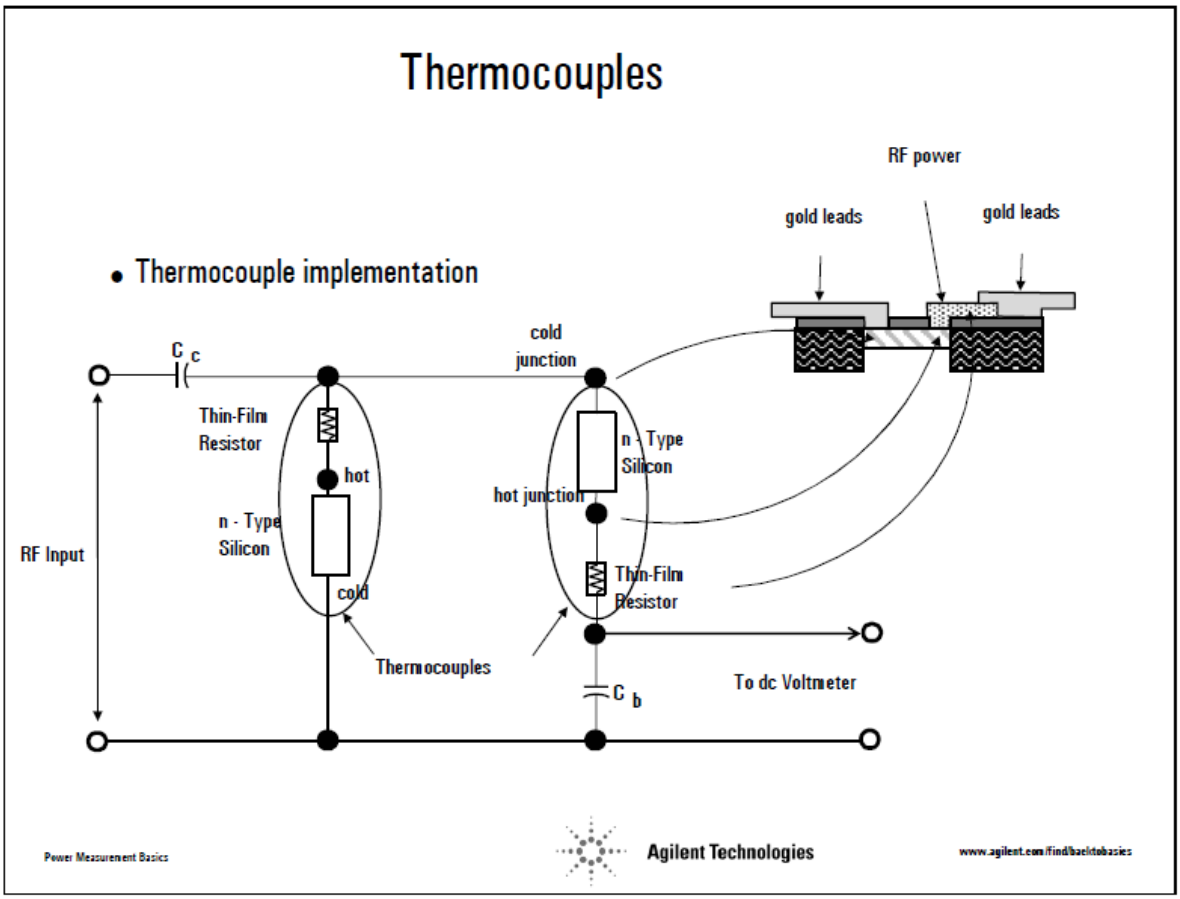


Thermocouple technology is the result of combining thin-film and semiconductor technologies to give a very accurate, rugged, and reproducible power sensor. Thermocouple sensors have been the detection technology of choice for sensing RF and microwave power since their introduction in 1974. The two main reasons for this are:
1) they operate over a wider power range
2) they are more rugged.
Since thermocouples, like thermistors, always respond to the true power of a signal, they are ideal for all types of signal formats from CW to complex digital phase modulations.

The above example shows what happens when a metal rod is heated at one end. As a result of increased thermal agitation, many additional electrons become free from their atoms on the left end. The increased free electron density on the left causes diffusion toward the right. Each electron migrating to the right leaves behind a positive ion. That ion

attracts the electron back to the left with a force given by Coulomb's Law. Equilibrium occurs when the rightward diffusion force equals the leftward force of Coulomb's law. The leftward force can be represented by an electric field pointing toward the right. The electric field gives rise to a voltage source.

Thermocouple sensors are based on the fact that a metal generates a voltage due to temperature differences between a hot and a cold junction and that different metals will create different voltages. A thermocouple is based on the idea of this difference in voltages between the two metals. If the two metals are put together in a closed circuit, current will flow due to the difference in the voltages. If the loop remains closed, current will flow as long as the two junctions remain at different temperatures. In a thermocouple, the loop is broken and a sensitive voltmeter is inserted to measure the net thermoelectric voltage of the loop. The voltage can be related to a temperature change which can be related to the increased temperature due to RF power incident upon the thermocouple element. Since the voltage produced in a thermocouple is on the order of microvolts, many pairs of junctions of thermocouples are connected in series so that the first junction of each pair is exposed to heat and the second junction is not. In this way the net voltage produced by one thermocouple adds to that of the next,and the next, and so on, yielding a larger thermoelectric output. Such a series connection of thermocouples is called a thermopile. This larger signal makes for simpler sensing circuitry.



One way to implement thermocouple technology to make power sensors is like the method shown above.The sensor contains two identical thermocouples on one chip, electrically connected as in the figure. For DC, the thermocouples are in series, while at RF frequencies they are in parallel. The two thermocouples in parallel form a 50 ohm termination for the RF transmission line.

Thermocouple measurements are open-loop, meaning an external reference source is necessary to match a particular sensor with its associated meter. The power reference is contained in the power meter. To verify the accuracy of the system, or adjust for a sensor of different sensitivity, the user connects the thermocouple sensor to the power-reference output and, using a calibration adjustment, sets the meter to display 1.00 mW. This calibration effectively transforms the system to a closed-loop substitution-type system, and provides confidence in traceability back to internal company standards or NIST standards.

2.6 Power Measurement History:

Since the late 1800s, when Nikola Tesla first demonstrated wireless transmission, there has been a need to measure the output of RF circuits. A major focus of Telsa's work was wireless transmission of electrical power, so he was often working in the megawatt range, and a relative indication of power was the discharge length of the "RF lightning" he produced. For obvious reasons, there was little incentive to attempt any sort of "contact" measurement!

Around 1888, an Austrian physicist named Ernst Lecher developed his "wires" technique as a method for measuring the frequency of an RF or microwave oscillator. The apparatus, often known as Lecher Wires, consisted of two parallel rods or wires, held a constant distance apart, with a sliding short circuit between them. The wires formed an RF transmission line, and by moving the shorting bar, Lecher could create standing waves in the line, resulting in a series of the peaks and nulls. By measuring the physical distance between two peaks or two nulls, the signal's wavelength in the transmission line, and thus its frequency could be calculated.

Initially, Lecher used a simple incandescent light bulb across the lines as power detector to locate the peaks and nulls. The apparent brightness of the bulb at the peaks also gave him a rough indication of the oscillator's output amplitude. One of the problems with using a bulb, however, was that the low (and variable) impedance of its filament changed the line's characteristics, and could affect the resonant frequency and output amplitude of the oscillator.

This was addressed by substituting a high-impedance, gas-discharge glow tube for the incandescent bulb. The glass tube was laid directly across the wires, and the field from a medium-voltage RF signal was adequate to excite a glow discharge in the gas tube. This didn't change the tank impedance as much, while keeping it easy to visually determine the peak and null locations as the tube was slid up and down the wires. Later, a neon bulb was used, but the higher striking voltage of neon made the nulls difficult to locate precisely.

In 1933, H.V. Noble, a Westinghouse engineer, refined some of Tesla's research, and was able to transmit several hundred watts at 100 MHz a distance of ten meters or so. This wireless RF power transmission was demonstrated at the Chicago World's Fair at the Westinghouse exhibit. His frequencies were low enough that the transmitted and received signal voltages could be directly measured by conventional electronic devices of the day – vacuum tube and cat's whisker detectors. At higher frequencies, however, these simple methods did not work as well – the tubes and cat's whiskers of the day simply lost rectification efficiency and repeatability.

The Varian brothers used another indicator technique in the late 1930s during their development of the Klystron. They drilled a small hole in the side of the resonant cavity and put a fluorescent screen next to it. A glow would indicate that the device was oscillating, and the brightness gave a very rough power indication as adjustments were made. In fact some small transmitters manufactured into the 1960s had a small incandescent or neon lamp in the final tank circuit for tuning. The tank was tuned for maximum lamp brightness. These techniques all fall more under the category of RF indicators than actual measurement instruments.

The water-flow calorimeter, a common device for other uses, was adapted for higher power RF measurements to measure the heating effect of RF energy, and found its way into use anywhere you could install a "dummy load." By monitoring flow of water and temperature rise as it cooled the load, it was simple to measure long-term average power dissipated by the load.

The thermocouple is one of the oldest ways of directly measuring low RF power levels. This is done by measuring its heating effect upon a load, and is still in common use today for the measurement of "true-RMS" power. Thermocouple RF ammeters have been in use since before 1930 but were restricted to the lower frequencies. It was not until the 1970s that thermocouples were developed that allowed their use as sensors in the VHF and Microwave range.

In later years, thermocouples and semiconductor diodes improved both in sensitivity and high-frequency ability. By the mid 1940s, the fragile, galena-based “cat-whisker” detectors were being replaced by stable, durable packaged diodes that could be calibrated against known standards, and used for more general-purpose RF power measurement.

Diode-based power measurement was further improved in the 50s and 60s, and Boonton Electronics made some notable contributions to the industry, initially in RF voltage measurement. The Model 91B was introduced in 1958 and could measure from below one millivolt to several volts. With a suitable termination, this yielded a calibrated dynamic range of about -50 dBm to +22 dBm over a frequency range of 200 kHz to 500 MHz.

RF voltmeters and power meters continued to evolve throughout the 70s with the application of digital and microprocessor technology, but these were all “average-only” instruments and few had any ability to quantify peak measurements. When a pulsed signal had to be characterized, the accepted technique was to use an oscilloscope and crystal detector to view the waveform in a qualitative fashion, and perform an average power measurement on the composite signal using either a CW power meter or a higher power measurement such as a calorimeter.

The “slideback wattmeter” used a diode detector, and substituted a DC voltage for the RF pulse while the pulse was off, giving a way to measure the pulse’s amplitude while compensating for duty cycle. However, a more common approach was to simply characterize a diode detector to correct for its pulse response – a technique pioneered by Boonton Radio, a local company that provided a great deal of technology to Boonton Electronics.

The modern realization of the peak power meter came into being in the early 1990s. Boonton Electronics, Hewlett Packard (later Agilent Technologies) and Wavetek all introduced instruments that were specifically designed to measure pulsed or modulated signals, and correct for nonlinear response of the detector diodes in real time. These

instruments have evolved over time with the application of better detectors and high-speed digital signal processing technology.

2.7 First practical approach :

Digital dc watt meter using pic microcontroller is used to measure dc power of dc circuits. Voltage and current sensors are used in this project. Voltage and current sensors are interfaced with microcontroller.

-Voltage sensor: is used to measure voltage across circuit or load.
-Current sensor: is used to measure current passing through load, Voltage divider circuit is used to measure high voltage. because microcontroller can not read high voltage or voltage more than 5 volt. Shunt resistor is used as a current sensor. Shunt resistor is used to convert current into voltage form. Because microcontroller does not understand current. Microcontroller can read voltage directly.

-Liquid crystal display (LCD): is used to show measured value of dc power.

-Circuit diagram of dc power meter: is given below. 0.47 resistor is used as a shunt resistor. Resistor R1 and R4 is used as a voltage divider**.** LCD displays measured DC power by fetching its value from PIC16F877A microcontroller. 8MHz crystal is used to operate PIC16F877A microcontroller. DC ammeter used in below circuit is just for simulation purpose

.

-DC power meter working: As we have mentioned above this projects consists of three main components. Details of these components is given below:

- **Digital Ammeter:** is designed using shunt resistor. R_6 0.46 Shunt resistor is used to measure current flowing through a load. Shunt resistor converts current passing through it into voltage. This voltage is measured with the help of analog channel AN_1 of PIC16F877A microcontroller. Measured voltage converted back

into current using ohm's law formula $I = V/R$. because value of shunt resistor and measured voltage is known.

- **Digital voltmeter:** is used to measure voltage across load. Voltage divider is used to step down voltage less than 5 volt. Voltage across R_4 resistor of voltage divider is measured with the help of of analog channel ANO of PIC16F877A microcontroller. Measured voltage converted back into actual voltage by multiplying it the opposite formula of voltage divider.
- **PIC16F877A microcontroller:** all mathematical calculations are done through programming of pic . it read current and voltages through ADC. As you know dc power is just a product of voltage and current.

DC power = Voltage * Current

Dc power meter code :

```
Sbit  LCD_RS at  RB4_bit;
Sbit  LCD_EN at  RB5_bit;
Sbit  LCD_D4 at  RB0_bit;
Sbit  LCD_D5 at  RB1_bit;
Sbit  LCD_D6 at  RB2_bit;
Sbit  LCD_D7 at  RB3_bit;

Sbit LCD_RS_Direction at TRISB4_bit;
Sbit LCD_EN_Direction at TRISB5_bit;
Sbit LCD_D4_Direction at TRISB0_bit;
Sbit LCD_D5_Direction at TRISB1_bit;
Sbit LCD_D6_Direction at TRISB2_bit;
Sbit LCD_D7_Direction at TRISB3_bit;
Float voltage, current, power ;
Char power [4];
Void main () {

        PORTA = 0XFF;
        TRISA  = 0XFF;
        PORTB = 0;
```

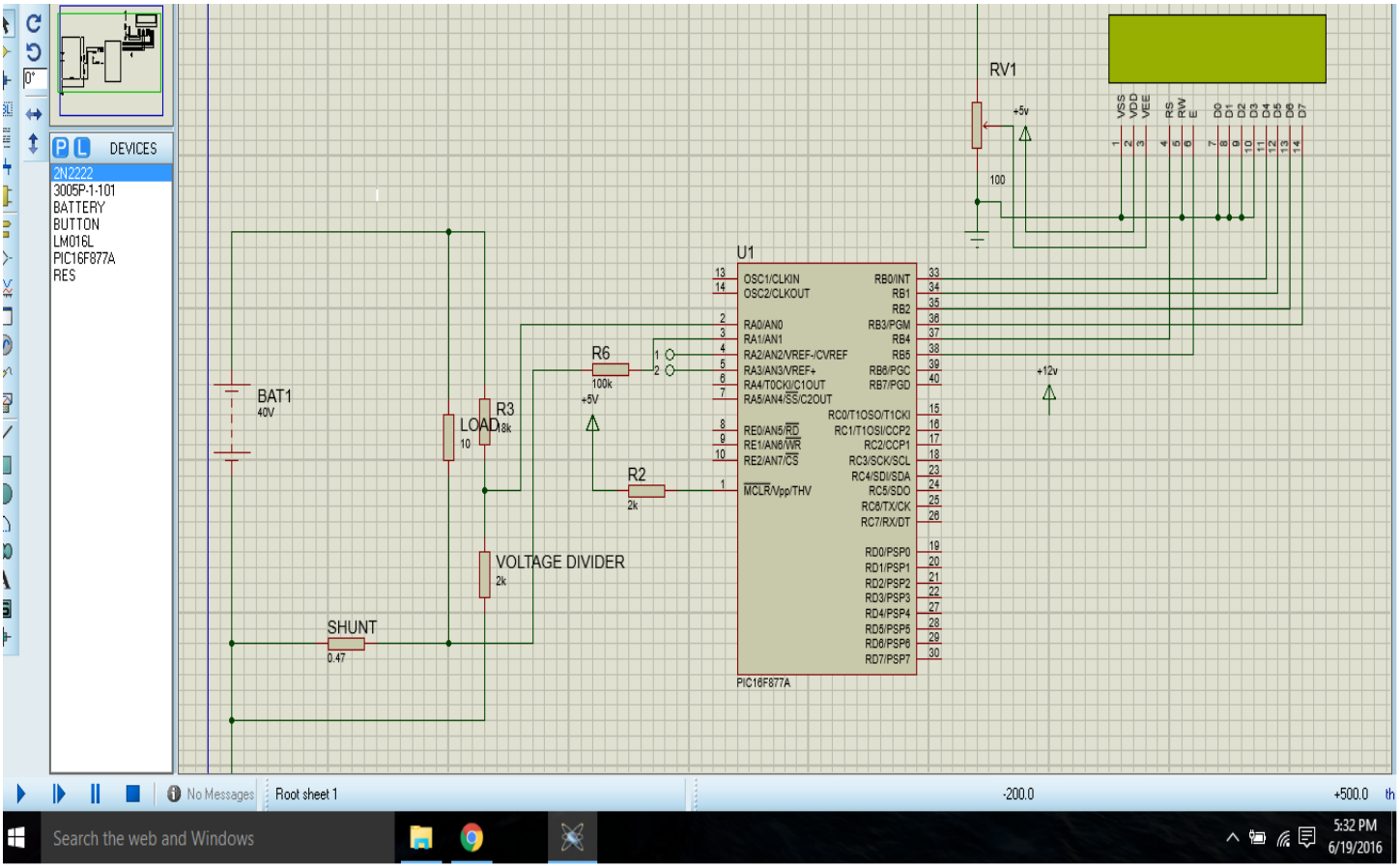
```
PORTA = 0;
LCD_init ();
ADC_init();
LCD_cmd(_LCD_CURSER_OFF);
LCD_cmd(_LCD_CLEAR);
LCD_out(1,1,”power meter’);
delay _ms(1000);


while(1)
{
        Voltage = ADC_Read(0);
        Voltage = (voltage*5*10) / 1024;
        Current = ADC_Read(1);
        Current = (current * 0.00489) / 0.047;
        Power = voltage * current ;
        inttostr(power,pwr);
        Lcd_out(2,1,”power= “ ;
        Lcd_out(2,8,Ltrim(pwr));
        Lcd_out(2,11, “W”) ;

        }
}
```

- Voltage = adc_read(0) : this function reads analog value of voltage and converts it into binary value of voltage.
- Voltage = (voltage*5*10) / (1024)) : it converts binary value back into actual voltage by multiplying it with ADC resolution factor and voltage divider inverse.
- Current = adc_read(1) ; this function reads analog value of voltage across shunt resistor.
- Current = (current * 0.00489) / 0.47) : this function converts binary value of voltage across shunt resistor into current.
- Power = voltage * current : it’s just a multiplication of measured voltage and measured current.
- inttostr(power,pwr) : it converts power value into string.

*Dc power meter Simulation :



Disadvantages of this practical approach :

1. Can't deal with very low power and very high power measurement .
2. Inaccurate voltage measuring .
3. Inaccurate current measuring due to losses in shunt resistor.
4. Very small range of power measurement.

Another way to perform ADC in power meter :

Using register programming instead of using built in library in mikroC programming, it is consist of Four main registers :

- **ADCON0 : A/D CONTROL REGISTER 0**

It consist of 7 bits and each of them have specific function as follows :

Bit 7 **unimplemented:** read as 0
Bit 6-2 **CHS<4:0>** Analog channel select bits
00000 = AN0
00001 = AN1
00010 = AN2
00011 = AN3
00100 = AN4
00101 = AN5
00110 = AN6
00111 = AN7
Bit 1 **GO/DONE:** A/D Conversion status bit
1 = A/D conversion cycle in progress.Setting this bit starts an A/D conversion cycle
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress.
Bit 0 **ADON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled and consumes no operating current.

- **ADCON1: A/D CONTROL REGISTER 1**

Also consist of 7 bits and each of them have specific function as follows :

Bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six Most Significant bits of ADRESH are set to '0' when the conversion result is loaded.
0 = Left justified. Six Least Significant bits of ADRESL are set to '0' when the
conversion result is loaded.
Bit 6-4 **ADCS<2:0>:** A/D Conversion Clock Select bits
000 =FOSC/2
001 =FOSC/8
010 =FOSC/32
011 =FRC (clock supplied from a dedicated RC oscillator)
100 =FOSC/4

101 =FOSC/16
110 =FOSC/64
111 =FRC (clock supplied from a dedicated RC oscillator)
Bit 3-2 **Unimplemented**: Read as '0'
Bit 1-0 **ADPREF<1:0>:** A/D Positive Voltage Reference Configuration bits

00 =VREF+ is connected to AVDD
01 = Reserved
10 =VREF+ is connected to external VREF+(1)
11 =VREF+ is connected to internal Fixed Voltage Reference (FVR) module(1).

- **ADRES Registers** : contain results of ADC

ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

Bit 7-2 **Reserved:** Do not use.
Bit 1-0 **ADRES<9:8>:** ADC Result Register bits Upper two bits of 10-bit conversion result.
ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1
bit 7-0 **ADRES<7:0>:** ADC Result Register bits Lower eight bits of 10-bit conversion result.

Chapter (3): Communication Techniques

Chapter 3
Communication Techniques

3.1 What is Serial?

The concept of serial communication is simple. The serial port sends and receives bytes of information one bit at a time. This is slower than parallel communication, which allows the transmission of an entire byte at once; however, it is simpler and can be used over longer distances. For example, the IEEE 488 specifications for parallel communication state that the cabling between equipment can be no more than 20 meters total, with no more than 2 meters between any two devices; serial can extend as much as 1200 meters.
Typically, serial is used to transmit ASCII data. Communication is completed using 3 transmission lines: (1) Ground, (2) Transmit, and (3) Receive. Since serial is asynchronous, the port is able to transmit data on one line while receiving data on another. This is referred to as Full-Duplex transmission. Other lines are available for handshaking, but are not required.

3.2 What are important serial characteristics?

They are baud rate, data bits, stop bits, and parity. For two ports to communicate, these parameters must match:

a. **Baud rate** is a speed measurement for communication. It indicates the number of bit transfers per second. For example, 300 baud is 300 bits per second. When we refer to a clock cycle, in the context of serial, we mean the baud rate. For example, if the protocol calls for a 4800 baud rate, then the clock is running at 4800Hz. This means that the serial port is sampling the data line at 4800Hz. Common baud rates for telephone lines are 14400, 28800, and 33600. Baud rates greater than these are possible, but these rates reduce the distance by which devices can be separated. These high baud rates are used for device communication where the devices are located near one another.

b. **Data bits** are a measurement of the actual data bits in a transmission. When the computer sends a packet of information, the amount of actual data may not be a full 8 bits. Standard values for the data packets are 5, 7, and 8 bits. Which setting you choose depends on what information you are transferring. For example, standard ASCII has values from 0 to 127 (7 bits). Extended ASCII uses 0 to 255 (8 bits). If the data being transferred is simple text (standard ASCII),

then sending 7 bits of data per packet is sufficient for communication. A packet refers to a single byte transfer, including start/stop bits, data bits, and parity. Since the number of actual bits depend on the protocol selected, the term packet is used to cover all instances.

c. **Stop bits** are used to signal the end of communication for a single packet. Typical values are 1, 1.5, and 2 bits. Since the data is clocked across the lines and each device has its own clock, it is possible for the two devices to become slightly out of sync. Therefore, the stop bits not only indicate the end of transmission but also give the computers some room for error in the clock speeds. The more bits that are used for stop bits, the greater the lenience in synchronizing the different clocks, but the slower the data transmission rate.

d. **Parity** is a simple form of error checking that is used in serial communication. There are four types of parity: even, odd, marked, and spaced. The option of using no parity is also available. For even and odd parity, the serial port will set the parity bit (the last bit after the data bits) to a value to ensure that the transmission has an even or odd number of logic high bits. For example, if the data was 011, then for even parity, the parity bit would be 0 to keep the number of logic high bits even. If the parity was odd, then the parity bit would be 1, resulting in 3 logic high bits. Marked and spaced parity does not actually check the data bits, but simply sets the parity bit high for marked parity or low for spaced parity. This allows the receiving device to know the state of a bit which enables the device to determine if noise is corrupting the data or if the transmitting and receiving devices' clocks are out of sync.

***To connect between meter circuits(Loads) of each home and the Master microcontroller(18f4620) we use serial communication protocol: RS-485**

3.3 What Is RS-485?

RS-485 (EIA-485 Standard) is an improvement over serial communication RS-422, because it increases the number of devices from 10 to 32 and defines the electrical characteristics necessary to ensure adequate signal voltages under maximum load. With this enhanced multi-drop capability, you can create networks of devices connected to a single RS-485 serial port. The noise immunity and multi-drop capability make RS-485 the serial connection of choice in industrial applications requiring many distributed devices networked to a PC or other controller for data

collection, HMI, or other operations. RS-485 is a superset of RS-422; thus, all RS-422 devices may be controlled by RS-485. RS-485 hardware may be used for serial communication with up to 4000 feet of cable. With Maximum Data Rate (at max cable length) equal 10 Mbit/s.

3.4 How does the hardware work?

Data is transmitted differentially on two wires twisted together, referred to as a "twisted pair." The properties of differential signals provide high noise immunity and long distance capabilities(with twisted-pair wire reduces two major sources of problems for designers of high-speed long-distance networks: radiated EMI and received EMI.). A 485 network can be configured two ways, "two-wire" or "four-wire." In a "two-wire" network the transmitter and receiver of each device are connected to a twisted pair. "Four-wire" networks have one master port with the transmitter connected to each of the "slave" receivers on one twisted pair. The "slave" transmitters are all connected to the "master" receiver on a second twisted pair. In either configuration, devices are addressable, allowing each node to be communicated to independently. Only one device can drive the line at a time, so drivers must be put into a high-impedance mode (tri-state) when they are not in use. Some RS-485 hardware handles this automatically. In other cases, the 485 device software must use a control line to handle the driver. A consequence of tri-stating the drivers is a delay between the end of a transmission and when the driver is tri-stated. This turn-around delay is an important part of a two-wire network because during that time no other transmissions can occur (not the case in a four-wire configuration). An ideal delay is the length of one character at the current baud rate (i.e. 1 ms at 9600 baud). The device manufacturer should be able to supply information on the delay for their products.

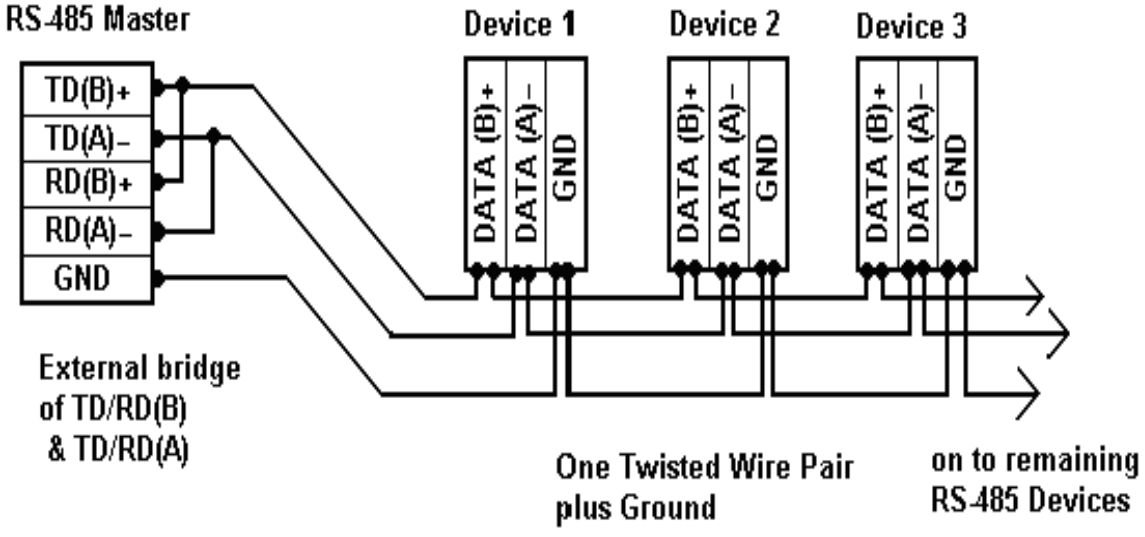
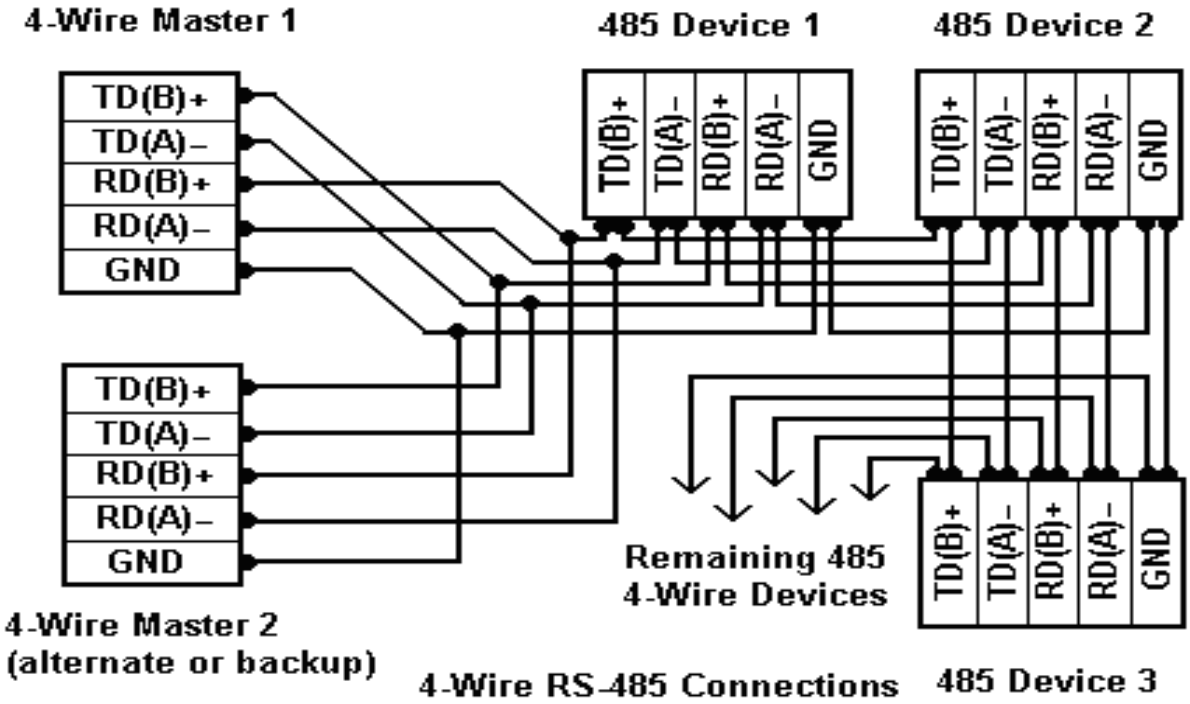


Fig. 3
2-Wire RS-485 Connections



Two-wire or four-wire? Two-wire 485 networks have the advantage of lower wiring costs and the ability for nodes to talk amongst themselves. On the downside, two-wire mode is limited to half-duplex and requires attention to turn-around delay. Four-wire networks allow full-duplex operation, but are limited to master-slave situations (i.e. a "master" node requests information from individual "slave" nodes). "Slave" nodes cannot communicate with each other. Remember when ordering your cable, "two-wire" is really two wires + ground, and "four-wire" is really four wires + ground.

3.5 How does the software work?

485 software handles addressing, turn-around delay, and possibly the driver tri-state features of 485. Determine before any purchase whether your software handles these features. Remember, too much or too little turn-around delay can cause troubleshooting fits, and delay should be a function of baud rate.

***RS-485 Library in mikroC PRO for PIC:**

RS-485 is a multipoint communication which allows multiple devices to be connected to a single bus. The mikroC PRO for PIC provides a set of library routines for work with RS485 system using Master/Slave

architecture. Master and Slave devices interchange packets of information. Each of these packets contains synchronization bytes, CRC byte, address byte and the data. Each Slave has unique address and receives only packets addressed to it. The Slave can never initiate communication.

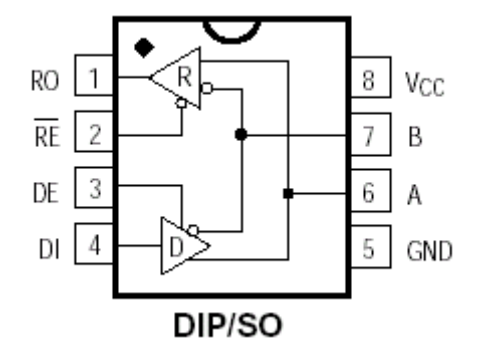
It is the user's responsibility to ensure that only one device transmits via 485 bus at a time.

The RS-485 routines require the UART module. Pins of UART need to be attached to RS-485 interface transceiver, such as LTC485(RS-485 interface) or similar.
In this project we use MAX487.

3.6 What is MAX487?

It is a low-power transceiver for RS-485, with these features:
-Low Quiescent current.
-(-7V) to (+12V) common-mode input voltage range.
-Three state output.
-Operates from a single 5V supply.
-Current limiting and thermal shutdown for driver overload protection.

Pin Layout



Pin Description

Pin Number	Description
1	RO - Receiver Output
2	RE - Receiver Output Enable
3	DE - Driver Output Enable
4	DI - Driver Input
5	GND - Ground
6	A - Non-Inverting Receiver Input and Driver Output
7	B - Inverting Receiver Input and Driver Output
8	Vcc - +5V Positive Supply

3.7 Applications among RS-485:

RS-485 signals are used in a wide range of computer and automation systems. In a computer system, SCSI-2 and SCSI-3 may use this specification to implement the physical layer for data transmission between a controller and a disk drive. RS-485 is used for low-speed data communications in commercial aircraft cabins vehicle bus. It requires minimal wiring, and can share the wiring among several seats, reducing weight.

RS-485 is used as the physical layer underlying many standard and proprietary automation protocols used to implement Industrial Control Systems.Utilizing a series of dedicated interface devices, it allows PCs and industrial controllers to communicate in a local area network utilizing a token passing medium access control.[6] These are used in programmable logic controllers and on factory floors. Since it is differential, it resists electromagnetic interference from motors and welding equipment.

In theatre and performance venues RS-485 networks are used to control lighting and other systems using the DMX512 protocol.

RS-485 is also used in building automation as the simple bus wiring and long cable length is ideal for joining remote devices. It may be used to

control video surveillance systems or to interconnect security control panels and devices such as access control card readers.

Although many applications use RS-485 signal levels; the speed, format, and protocol of the data transmission is not specified by RS-485. Interoperability of even similar devices from different manufacturers is not assured by compliance with the signal levels alone.

3.8 Code Examples.1 : "using mikroC PRO for PIC"

Ex1:

***RS-485 Master code(18452):**

```
LCD module connections //

sbit LCD_RS at RB4_bit

sbit LCD_EN at RB5_bit

sbit LCD_D4 at RB0_bit

sbit LCD_D5 at RB1_bit

sbit LCD_D6 at RB2_bit

sbit LCD_D7 at RB3_bit


sbit LCD_RS_Direction at TRISB4_bit

sbit LCD_EN_Direction at TRISB5_bit

sbit LCD_D4_Direction at TRISB0_bit

sbit LCD_D5_Direction at TRISB1_bit

sbit LCD_D6_Direction at TRISB2_bit

sbit LCD_D7_Direction at TRISB3_bit

End LCD module connections //
```

```
sbit  rs485_rxtx_pin  at RC2_bit;              // set transcieve pin

sbit  rs485_rxtx_pin_direction at TRISC2_bit;   // set transcieve pin
direction

var//

꞉int pwr=0

꞉[۷]char pwrtxt

꞉[۹]char dat

modules//

}()void recieve

}if(uart1_data_ready())

꞉()pwr=uart1_read

꞉inttostr(pwr,pwrtxt)

꞉lcd_out(1,1,"recieve=")

꞉lcd_out_cp(pwrtxt)

if//{

void//{

}()void main

config//

꞉portC=0x00

trisC=0xff; //input

꞉portB=0x00

trisB=0x00; //output

init//

UART1_Init(9600);                  // initialize UART1 module
```

ؼ(١٠٠)Delay_ms

RS485Master_Init(); // initialize MCU as Master

ؼdat[0] = 65//

ؼdat[1] = 0//

ؼdat[2] = 0//

dat[4] = 0; // ensure that message received flag is 0//

dat[5] = 0; // ensure that error flag is 0//

ؼdat[6] = 0//

RS485Master_Send(dat,1,160)//

ؼ()lcd_init

ؼlcd_cmd(_lcd_cursor_off)

while1//

}(١)while

ؼ()recieve

{

}

***RS-485 Slave Code(12f1822):**

sbit RS485_rxtx_pin at RA2_bit; // transmit/receive control set to PORTC.B2

sbit RS485_rxtx_pin_direction at TRISA2_bit; // RxTx pin direction set as output

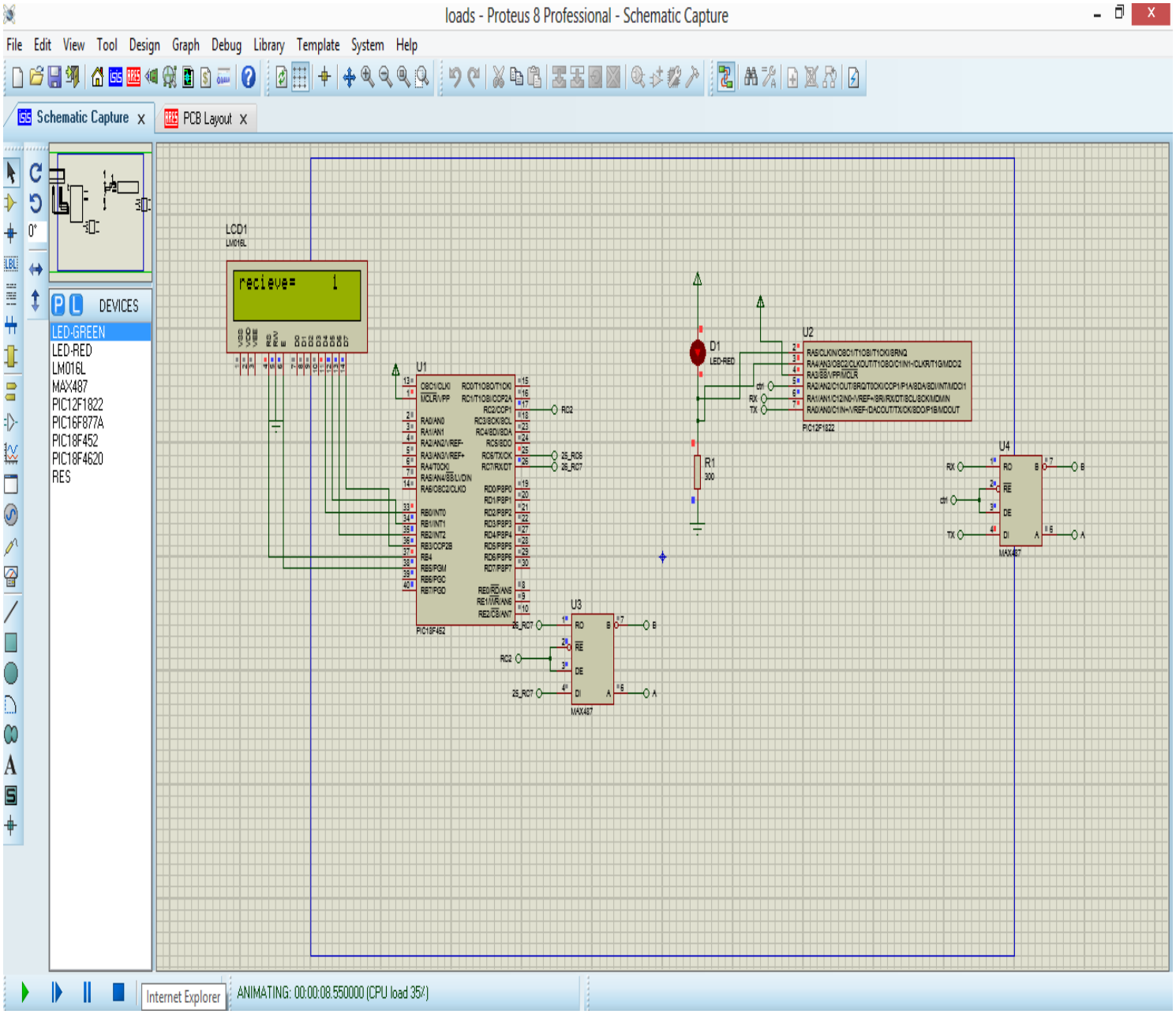
ؼfloat power

ؼfloat current

ؼfloat voltage

```
؛[٩]char dat
} ()void main
UART1_Init(9600);               // Initialize UART module at 9600 bps
؛(١٠٠)Delay_ms
RS485Slave_Init(4);               // intialize MCU as a Slave for RS-485 communication with address 4
؛(٢٠)delay_us
}(١)while
meter's code//
؛(٤)voltage = ADC_Read
؛(٥)current = ADC_Read
؛voltage = (voltage*5)/1023
؛power = voltage*current
؛FloatToStr(power,dat )
////////
؛RS485Slave_Send(dat, 1)
؛(١٠٠٠)delay_ms
{
{
```

*Simulation: "using Proteus"



Ex2:

*RS-485 Master Code(18f452):

```
sbit  rs485_rxtx_pin  at RC5_bit;              // set transcieve pin

sbit  rs485_rxtx_pin_direction at TRISC5_bit;   // set transcieve pin
direction

var//
```

```
unsigned int rec[7]; //received data from 16f
unsigned int trans; /*transmitted data(from Raspberry)to
/*.f using I2C, that includes specific code١٦
؛unsigned int state1
؛[٢]unsigned int pwr1
؛unsigned int state2
؛[٢]unsigned int pwr2
؛unsigned int batt_chg
unsigned int tot_pwr[2];//total power in home
unsigned int cond; //condition: Normal (no distribution to power)
.unsigned int dat[14]; //array that includes power and states of two loads
؛int i
؛[٢]unsigned int pwr_ex
؟؟؟؟؟؟؟؟؟؟؟؟const Add=0x12;//what is the address of 18f
//////////////////////////////////////////////////////
()void recieve
}
}(١)while
}if (trans=12)
؛RS485Master_Receive(rec)
؛ [٠]pwr1[0]= rec
؛ [١]pwr1[1]= rec
؛ [٢]pwr1[2]= rec
؛rec[4]=0
؛rec[5]=0
؛rec[6]=0
؛(٢٠٠)delay_ms
```

؛RS485Master_Receive(rec)

؛[٠]pwr1[3]= rec

؛[١]pwr1[4]= rec

؛[٢]state1= rec

؛rec[4]=0

؛rec[5]=0

؛rec[6]=0

{

}if (trans=22)

؛RS485Master_Receive(rec)

؛ [٠]pwr2[0]= rec

؛ [١]pwr2[1]= rec

؛ [٢]pwr2[2]= rec

؛rec[4]=0

؛rec[5]=0

؛rec[6]=0

؛(٢٠٠)delay_ms

؛RS485Master_Receive(rec)

؛[٠]pwr2[3]= rec

؛[١]pwr2[4]= rec

؛[٢]state2= rec

؛rec[4]=0

؛rec[5]=0

؛rec[6]=0

{

؛dat[0]=batt_chg

؛[٠]dat[1]=tot_pwr

؛[١]dat[2]=tot_pwr

؛[٢]dat[3]=tot_pwr

؛[٣]dat[4]=tot_pwr

؛[٤]dat[5]=tot_pwr

؛ dat[6]=cond

؛[٠]dat[7]=pwr_ex

؛[١]dat[8]=pwr_ex

؛[٢]dat[9]=pwr_ex

؛[٣]dat[10]=pwr_ex

؛[٤]dat[11]=pwr_ex

؛[٠]dat[12]=pwr1

؛[١]dat[13]=pwr1

؛[٢]dat[14]=pwr1

؛[٣]dat[15]=pwr1

؛[٤]dat[16]=pwr1

؛dat[17]=state1

؛[٠]dat[18]=pwr2

؛[١]dat[19]=pwr2

؛[٢]dat[20]=pwr2

؛[٣]dat[21]=pwr2

؛[٤]dat[22]=pwr2

؛dat[23]=state2

{

{

}()void main

config//

؛portC=0x00*/

```
trisC=0x00; //output
꞉portB=0x00
trisB=0x00; //output
꞉SSPADD =  Add
꞉SSPCON1=0b00101000//
꞉SSPCON2=0b01001111//
/*꞉SSPSTAT=0b01000101//
I2C1_Init(100000);        // initialize I2C communication
I2C1_Start();             // issue I2C start signal
}(١)while
trans=I2C1_Rd(0);           // rec.data from Rasp. by I2C
{
I2C1_Stop();              // issue I2C stop signal
꞉(٥٠٠)delay_ms
UART1_Init(1202);                 // initialize UART1 module
꞉(١٠٠)Delay_ms
RS485Master_Init();               // initialize 18f as Master
RS485Master_Send(trans,1,12);//12=address of home
꞉()recieve
꞉(٥٠٠)delay_ms
꞉(١٠٠٠٠٠)I2C1_Init
꞉()I2C1_Start
꞉I2C1_wr(Add)//
}(١)while
for (i=0; i<=12; i++);/*for sending "dat" array byte by byte
/*for Rasp by I2C
}
```

{

؛I2C1_Wr(dat)

{

؛()I2C1_Stop

{

***RS-485 slave Code(16f877A):**

sbit RS485_rxtx_pin at RA2_bit; // transmit/receive control set to PORTC.B2

sbit RS485_rxtx_pin_direction at TRISA2_bit; // RxTx pin direction set as output

؛unsigned int m_pwr1

؛unsigned int m_pwr2

؛[٥]char m_pwr1_txt

؛[٥]char m_pwr2_txt

؛float c1

؛float c2

؛float v1

؛float v2

؛[٧]unsigned int rec

؛[٧]unsigned int dat1

؛[٧]unsigned int dat2

unsigned int state1; // to indicate state of load: off=0, on=1

؛unsigned int state2

}()void trans1

meter's code//

؛(١٠٢٣/٢١,٥)*(v1= adc_read(0

؛(١٠٢٣/٥,٠)*(c1= adc_read(1

؛m_pwr1= v1*c1*1000

```
؛inttostr(m_pwr1,m_pwr1_txt)
؛[٠]dat1[0]=m_pwr1_txt
؛[١]dat1[1]=m_pwr1_txt
؛[٢]dat1[2]=m_pwr1_txt
؛dat1[3]= 3
؛dat1[4]=0
؛dat1[5]=0
؛ RS485Slave_Send(dat1, 3)
؛(٥٠٠)delay_ms
؛[٣]dat1[0]=m_pwr1_txt
؛[٤]dat1[1]=m_pwr1_txt
؛dat1[2]=state1
؛dat1[3]= 3
؛dat1[4]=0
؛dat1[5]=0
؛ RS485Slave_Send(dat1, 3)
؛(٥٠٠)delay_ms
{
}()void trans2
meter's code//
؛(١٠٢٣/٢١,٥)*(v2= adc_read(0
؛(١٠٢٣/٥,٠)*(c2= adc_read(1
؛m_pwr2= v2*c2*1000
؛inttostr(m_pwr2,m_pwr2_txt)
/////
؛[٠]dat2[0]=m_pwr2_txt
؛[١]dat2[1]=m_pwr2_txt
```

```
؛[٢]dat2[2]=m_pwr2_txt
؛dat2[3]= 3
؛dat2[4]=0
؛dat2[5]=0
؛ RS485Slave_Send(dat2, 3)
؛(٥٠٠)delay_ms
؛[٣]dat2[0]=m_pwr2_txt
؛[٤]dat2[1]=m_pwr2_txt
؛dat2[2]=state2
؛dat2[3]= 3
؛dat2[4]=0
؛dat2[5]=0
؛ RS485Slave_Send(dat2, 3)
؛(٥٠٠)delay_ms
{
} ()void main
؛porta=0x00
trisa=0xff; //pins of loads
؛portb=0x00
trisb=0x00;//pins for load control
UART1_Init(1202);              // Initialize UART module at 9600 bps
؛(١٠٠)Delay_ms

RS485Slave_Init(12);              // intialize MCU as a Slave for RS-485 communication with address 4
؛(١٠)delay_ms
}(١)while
؛RS485Slave_Receive(rec)
```

```
}if(rec[0]==12)
꞉()trans1
{
}if(rec[0]==22)
꞉()trans2
{
}if(rec[0]==11)
portb.b1=0; //make load1 on(active low)
꞉'state1='O
{
}if(rec[0]==10)
portb.b1=1; //make load1 off(active low)
꞉'state1='f
{
}if(rec[0]==20)
portb.b2=1; //make load2 off(active low)
{                       ꞉'state2='f
}if(rec[0]==21)
portb.b2=0; //make load2 on(active low)
{                       ꞉'state2='O

{
{
```

3.9 What is I2C?

I²C is a multi-master protocol that uses 2 signal lines. The two I²C signals are called ‘serial data’ (SDA) and ‘serial clock’ (SCL). Virtually any number of slaves and any number of masters can be connected onto these 2 signal lines and communicate between each other using a protocol that defines:

– 7-bits slave addresses: each device connected to the bus has got such a unique address;
– data divided into 8-bit bytes
– a few control bits for controlling the communication start, end, direction and for an acknowledgment mechanism.

The data rate has to be chosen between 100 kbps, 400 kbps and 3.4 Mbps, respectively called standard mode, fast mode and high speed mode. Some I²C variants include 10 kbps (low speed mode) and 1 Mbps (fast mode +) as valid speeds.

Physically, the I²C bus consists of the 2 active wires SDA and SCL and a ground connection (refer to figure 4). The active wires are both bi-directional. The I2C protocol specification states that the IC that initiates a data transfer on the bus is considered the Bus Master. Consequently, at that time, all the other ICs are regarded to be Bus Slaves.

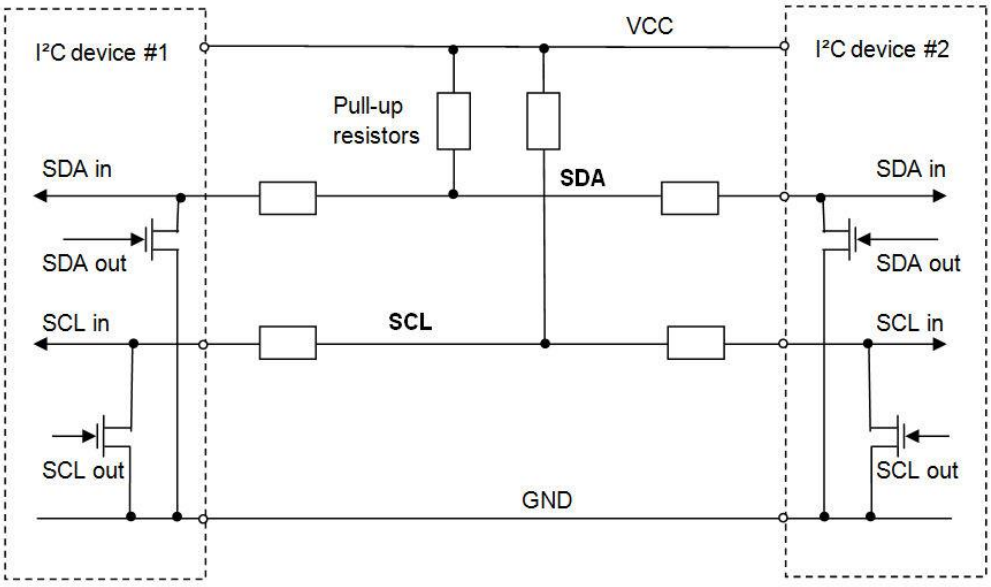


Figure 4: I²C bus with 2 devices connected. SDA and SCL are connected to VCC through pull-up resistors. Each device controls the bus lines outputs with open drain buffers.

3.10 What is Operation of I2C?

At the physical layer, both SCL and SDA lines are open-drain I/Os with pull-up resistors (refer to figure 4). Pulling such a line to ground is decoded as a logical zero, while releasing the line and letting it flow is a logical one. Actually, a device on a I²C bus 'only drives zeros'.

First, the master will issue a START condition. This acts as an 'Attention' signal to all of the connected devices. All ICs on the bus will listen to the bus for incoming data.

Then the master sends the ADDRESS of the device it wants to access, along with an indication whether the access is a Read or Write operation. Having received the address, all IC's will compare it with their own address. If it doesn't match, they simply wait until the bus is released by the stop condition (see below). If the address matches, however, the chip will produce a response called the ACKNOWLEDGE signal.

Once the master receives the acknowledge, it can start transmitting or receiving DATA. When all is done, the master will issue the STOP condition. This is a signal that states the bus has been released and that the connected ICs may expect another transmission to start any moment.

When a master wants to receive data from a slave, it proceeds the same way, but sets the RD/nWR bit at a logical one. Once the slave has acknowledged the address, it starts sending the requested data, byte by byte. After each data byte, it is up to the master to acknowledge the received data.

3.11 Why I2C?

- Independent Master, Slave, and Monitor functions.
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple I2C slave addresses supported in hardware.
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I ^ 2C bus addresses.
- 10-bit addressing supported with software assist.
- Supports SMBus.
- Only two bus lines are required

- No strict baud rate requirements, the master generates a bus clock
- I2C is a true multi-master bus providing arbitration and collision detection

3.12 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE "I2C Mode" in 18f4620 microcontroller:

The MSSP module in I2C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing. Two pins are used for data transfer:

• Serial clock (SCL) – RC3/SCK/SCL
• Serial data (SDA) – RC4/SDI/SDA The user must configure these pins ..as inputs or outputs through the TRISC<4:3> bits

–REGISTERS:

The MSSP module has six registers for I2C operation. These are:

• MSSP Control Register 1 (SSPCON1)
• MSSP Control Register 2 (SSPCON2)
• MSSP Status Register (SSPSTAT)
• Serial Receive/Transmit Buffer Register (SSPBUF)
• MSSP Shift Register (SSPSR) – Not directly accessible
• MSSP Address Register (SSPADD) SSPCON1, SSPCON2 and SSPSTAT are the control and status registers in I2C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write. SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from. SSPADD register holds the slave device address when the MSSP is configured in I2C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPADD act as the Baud Rate Generator reload value. In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set. During transmission, the SSPBUF is not double- buffered. .A write to SSPBUF will write to both SSPBUF and SSPSR

3.13 Code Examples.2 : "using mikroC PRO for PIC"

***I2C Master Code(18452):**

```
؛int var
؛char byte
؛const Add=0x69
} ()void main
؛portc=0x00
؛trisc=0x00
؛porta=0x00
؛trisa=0x00
؛SSPCON1=0b00101000
؛SSPCON2=0b01001111
؛SSPSTAT=0b01000101
I2C1_Init(100000);        // initialize I2C communication
I2C1_Start();             // issue I2C start signal
I2C1_Wr(Add);  //address of the slave
I2C1_Wr(0xff);            // send data (data to be written)
I2C1_Stop();              // issue I2C stop signal
؛(٣٠٠٠)delay_ms
I2C1_Init(100000);        // initialize I2C communication
I2C1_Start();             // issue I2C start signal
؛I2C1_Wr(Add)//
var = I2C1_Rd(0);     //0 == No Acknowledgement
I2C1_Stop();              // issue I2C stop signal
}(١)while
if(var == 0xff)
}
```

꞉porta = 0xff

{

{

{

***I2C slave Code(16f877A):**

꞉int byte

const Add = 0x69; // set I2C device address

}()void main

꞉portc=0x00

꞉trisc=0xff

꞉porta=0x00

꞉trisa=0x00

SSPADD = Add; // Get address (7bit). Lsb is read/write flag

SSPCON1=0b00101110; // Set to I2C slave with 7-bit //
address

꞉SSPCON2=0b10000001//

꞉SSPSTAT=0b01000100//

PIE1.SSPIF = 1; // enable SSP interrupts//

I2C1_Init(100000); // initialize I2C communication

I2C1_Start(); // issue I2C start signal

byte = I2C1_Rd(0); //0 == No Acknowledgement

I2C1_Stop(); // issue I2C stop signal

}(ꞏ)while

if(byte == 0xff)

}

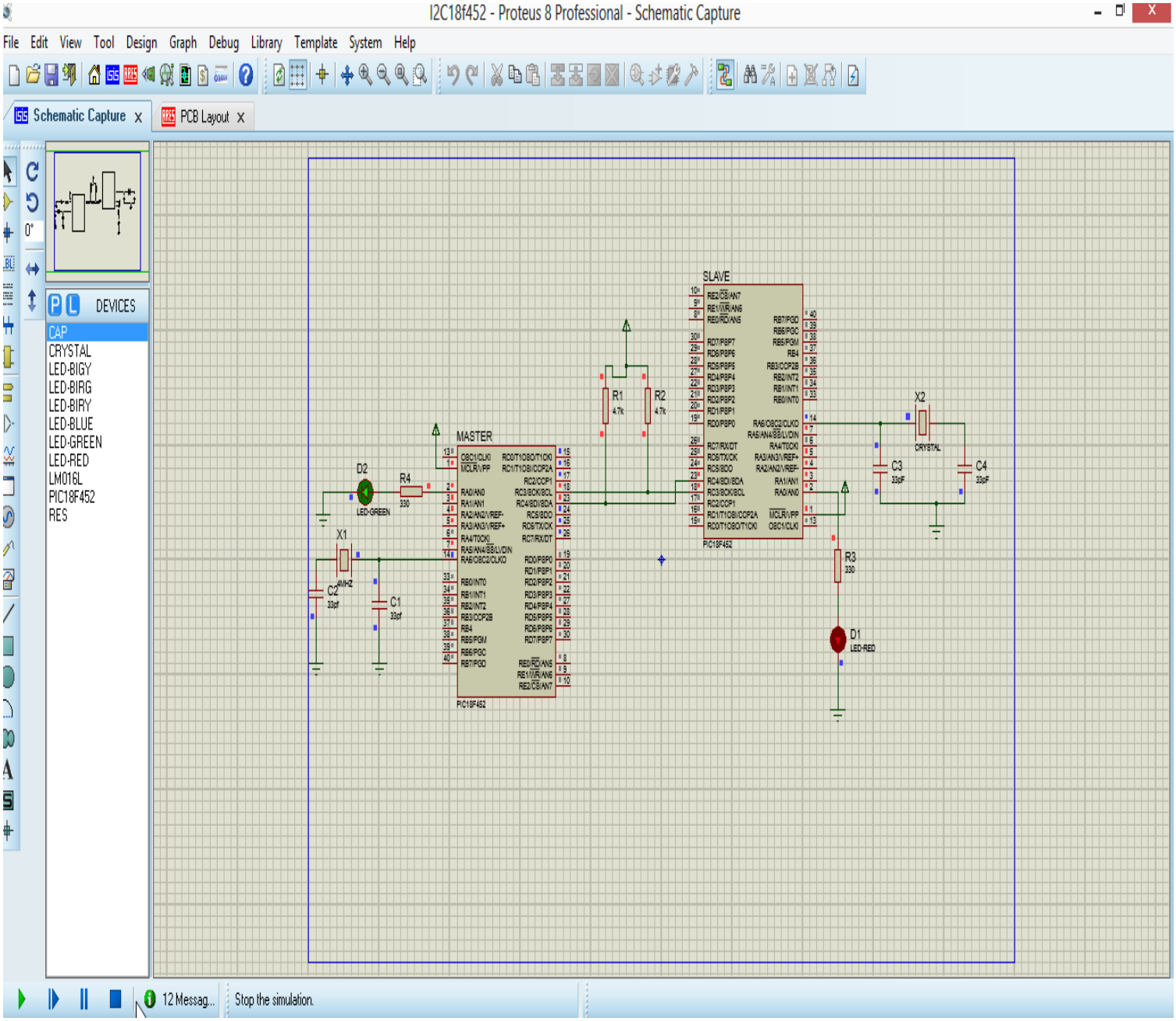
꞉porta = 0xff

{

I2C1_Init(100000); // initialize I2C communication

```
I2C1_Start();              // issue I2C start signal
꞉I2C1_wr(Add)//
I2C1_Wr(byte);             // send data (data to be written)
I2C1_Stop();               // issue I2C stop signal
{
```

***Simulation: "using Proteus"**



Chapter (4): Raspberry Pi

Chapter 4
Raspberry Pi

4.1 What is the Raspberry Pi?

The Raspberry Pi is a small computer about the size of a credit card and costs approximately £25. It was developed in the UK by the Raspberry Pi Foundation with the hope of inspiring a generation of learners to be creative and to discover how computers are programmed and how they function.

This small computer features amazing HD (high-definition) quality video playback, sports high quality audio and has the ability to play 3D games. The device uses the ARM processor which does most of the hard work in order to run the Raspberry Pi. ARM processors can be thought of as the brains of the device.

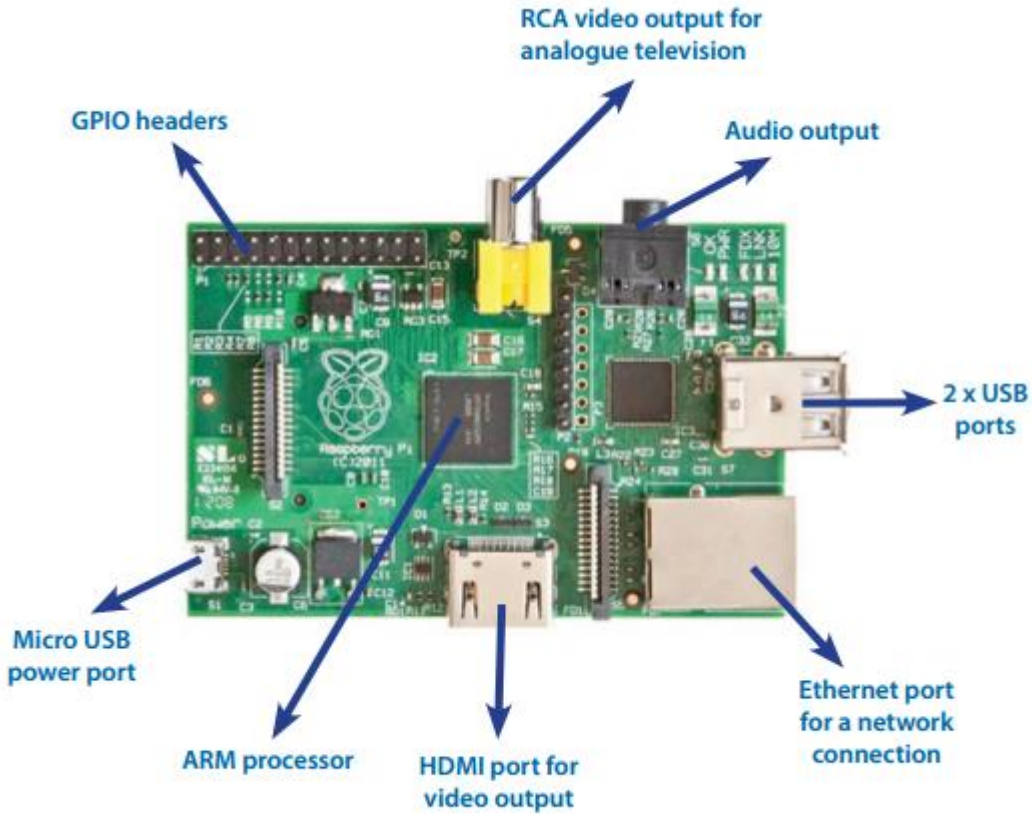
These processors are mainly used in small devices such as mobile phones, hand held mobile gaming devices and other small digital devices. The reason for this is that ARM processors are extremely efficient and fast when used in small devices. This makes the ARM processor the obvious choice for the Raspberry Pi.

Even though the Raspberry Pi is a computer it does not have a hard drive like traditional computers, instead it relies on the SD card for the starting up and storing of information. For the Raspberry Pi the SD card does the same job as a hard drive does in a traditional computer.

The SD card must contain the operating system, programs and the data needed to run the Raspberry Pi. The operating system tells the Raspberry Pi how to function, how to handle any input from the user and how to manage programs when they are running.

4.2 Hardware:

Hardware is a physical device that can be touched or held, like a hard drive or a mobile phone. Software can be thought of as a program or a collection of programs that instruct a computer on what to do and how to do it. Below is an image of the Raspberry Pi which describes some of the components that make up the hardware.



***Micro USB power port**

The micro USB power port is used to power the Raspberry Pi device.

***HDMI port**

The HDMI output is used to plug into a modern television or monitor.

***Ethernet port**

The Ethernet port is used to connect the Raspberry Pi to the internet or a local network.

***USB ports**

USB 2.0 ports are used to plug in a keyboard, mouse, web cam, external hubs etc.

***Audio output**

The audio output can be used to plug into an external amplifier or an audio docking station.

***GPIO headers**

The GPIO headers are used to connect the Raspberry Pi to other hardware devices. For example, they can be used to connect to LEDs, motors and other electronic components.

***RCA video output**

The video output is used to connect to an older type television.

***ARM processor**

The ARM processor can be thought of as the brains of the Raspberry Pi.

4.3 Ingredients :

1 x SD Card - 4GB or above

1 x Monitor or TV with a HDMI cable (or RCA cable if it is an analogue TV)

Note: Some monitors do not have HDMI ports – in this situation you will need to have an adapter – you can get a HDMI to DVI or a HDMI to VGA.

1 x Mouse

1 x Keyboard

1 x Ethernet LAN cable (optional - needed if you require a network connection)

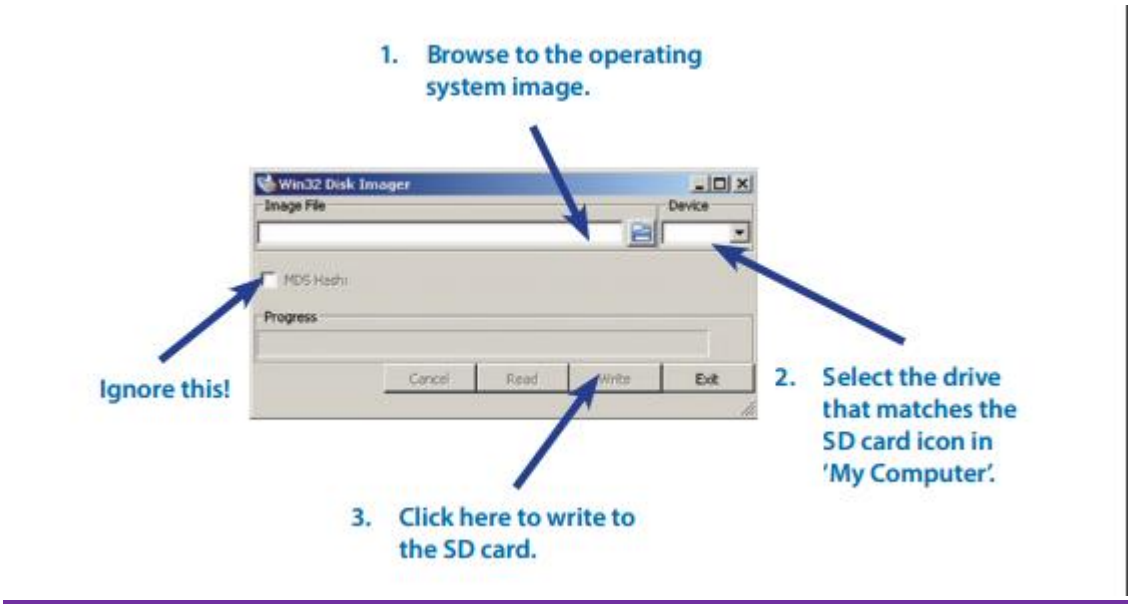
1 x Micro USB power supply (many mobile phone chargers will work but it must supply 5V and have at least 700mA)

4.4 Preparing the Raspberry Pi :

Before you can connect and power up the Raspberry Pi you will have to create a working operating system. In order to install the operating system you will need a blank SD card. It is recommended that the SD is at least 4 GB in size. You can also use a larger SD card if you intend to store a lot

of data. If you will be using a pre-loaded SD card then you can skip over this tutorial. Installing Raspberry Pi using Windows XP / Vista / 7 / 8 You can download an operating system from the official Raspberry Pi website www.raspberrypi.org/downloads - the recommended operating system is Raspbian "wheezy". If you are using Microsoft Windows you will need to save your downloaded operating system image to a folder and extract the contents. This will create a file that ends with a 'xxxx.img' file extension. The file name may vary depending on which version is available at the time. The main point is that you have a file ending in 'img'. In order to write this image (.img) file to the SD card you will need a program to do this. Download the following program called 'Win32 Disk Imager' and extract it. The download can be found by visiting www.raspberryshake.com/preparingraspberry-pi and clicking the 'Download here' link.

When the program is opened it will be expecting the image file which is the operating system image. It also requires a device to write to. You can ignore the MD5 hash check box as we want to concentrate on making this installation as easy as possible.



Click on the blue folder icon and browse to the image file. Select your SD card device letter by clicking on the device drop down. Check that you are writing to the SD card device. If you have any doubt you can double

click on My Computer and see that the drive letter that has the SD icon next to it. When you are sure that you have selected the correct image file and the correct device to write to, click on the write button and wait for the progress bar to reach 100%. When this has completed, safely eject your SD card. Make sure your Raspberry Pi is powered off and plug your SD card into it.

4.5 Basic Networking

Computer networks are made up of a group of computers or devices. Each computer requires a unique address in order to know where to send data to. When a letter is sent in the post it will contain the name of the person, the house number and the area they are located in. This is the same when data is sent over the internet. An example of an address looks like this: 192.168.100.1 This type of internet addressing is called IPv4 and stands for Internet Protocol version 4. The problem with IPv4 has become clear since more and more devices are being connected to the internet.

Each device that connects to the internet needs an address and IPv4 has a limited amount of addresses available. The answer to this was to create a new version called IPv6 which allows for more devices to be connected. An example of IPv6 looks like this: 2001:0db8:85a3:0042:0000:8a2e:0370:7334 A network consists of client computers and servers each having a unique internet address. Clients will request information from the server and the server returns this information. When you are visiting a web site your web browser is the client and the web site host is the server. When a client asks the server for this information it is called a client request

Servers listen for client requests using a port. A port is like a door waiting for someone to knock on it. When it receives a request from a client the server will respond with the information. Each port has a number which determines the type of data to be sent. Some ports are reserved so that when a client sends a request for information, it always uses the same port number. For example when you enter an address into your Raspberry Pi's web browser, it will connect to port 80 which is reserved for delivering web pages. The browser takes care of which port number to use so that you don't have to concern yourself with this. You can view the following file which lists your network interfaces:

/etc/network/interfaces

A snippet of the file looks like this:

iface eth0 inet dhcp

The word iface is referring to an interface and eth0 is referring to the Ethernet port. Eth0 is the only Ethernet interface on the Raspberry Pi. The rest of the line uses something called DHCP (dynamic host configuration protocol) a protocol used to configure network devices. DHCP is used to automatically obtain a unique IP address which allows your Raspberry Pi to be connected to a network.

4.6 An Introduction to Python

Flexible and powerful, Python was originally developed in the late 1980sat the National Research Institute for Mathematics and Computer Science by Guido van Rossum as a successor to the ABC language. Since its introduction, Python has grown in popularity thanks to what is seen as a clear and expressive syntax developed with a focus on ensuring that code is readable. Python is a high-level language. This means that Python code is written in largely recognizable English, providing the Piwith commands in a manner that is quick to learn and easy to follow. This is in marked contrast to low-level languages, like assembler, which a recloser to how the computer "thinks" but almost impossible for a human to follow without experience. The high-level nature and clear syntax of Python make it a valuable tool for any one who wants to learn to program.

4.7 Raspberry Pi role in our project :

It can be considered as control broker between homes and server.

It receives data from microcontroller through I2C communication protocol.

Server can control turning on and off any load in each home through it, as raspberry pi send specific signal for each order.

As home 1 has address 100 , when raspberry send “10 “ this means that Microcontroller will turn off load 1 in home 1 .

When raspberry send “11 “ this means that microcontroller will turn on load 1 in home 1 and so ,,

_While programming we made many stores on the raspberry like:

Battery charge

Power at home 1, 2

Status of each load “on, off “

It has a real important role in this project.

Chapter (5): Server

Chapter 5
Server

5.1 Server definition :

In computing, a **server** is a computer program or a device that provides functionality for other programs or devices, called "clients". Thisarchitecture is called the client–server model, and a single overall computation is distributed across multiple processes or devices. Servers can provide various functionalities, often called "services", such as sharing data or resources among multiple clients, or performing computation for a client. A single server can serve multiple clients, and a single client can use multiple servers. A client process may run on the same device or may connect over a network to a server on a different device.[1] Typical servers are database servers, file servers,mail servers, print servers, web servers, game servers, and application servers. Client–server systems are today most frequently implemented by (and often identified with) the request–response model: a client sends a request to the server, which performs some action and sends a response back to the client, typically with a result or acknowledgement. Designating a computer as "server-class hardware" implies that it is specialized for running servers on it. This often implies that it is more powerful and reliable than standard personal computers, but alternatively, large computing clusters may be composed of many relatively simple, replaceable server components.

5.2 Server's operating systems :

1)Windows

2)Linux

3)Unix

- GUI not available or optional
- Ability to reconfigure and update both hardware and software to some extent without restart
- Advanced backup facilities to permit regular and frequent online backups of critical data,
- Transparent data transfer between different volumes or devices
- Flexible and advanced networking capabilities

- Automation capabilities such as daemons in UNIX and services in Windows
- Tight system security, with advanced user, resource, data, and memory protection.
- Advanced detection and alerting on conditions such as overheating, processor and disk failure.

In practice, today many desktop and server operating systems share similar code bases, differing mostly in configuration.

The main purpose of any server is:
The purpose of a server is to share data as well as to share resources and distribute work. A server computer can serve its own computer programs as well; depending on the scenario, this could be part of a *quid pro quo* transaction, or simply a technical possibility. The following table shows several scenarios in which a server is used.

5.3 Hardware Requirements:

The purpose of a server is to share data as well as to share resources and distribute work. A server computer can serve its own computer programs as well; depending on the scenario, this could be part of a *quid pro quo* transaction, or simply a technical possibility. The following table shows several scenarios in which a server is used.

A rack-mountable server with the top cover removed to reveal internal components

5.4 Large Servers :

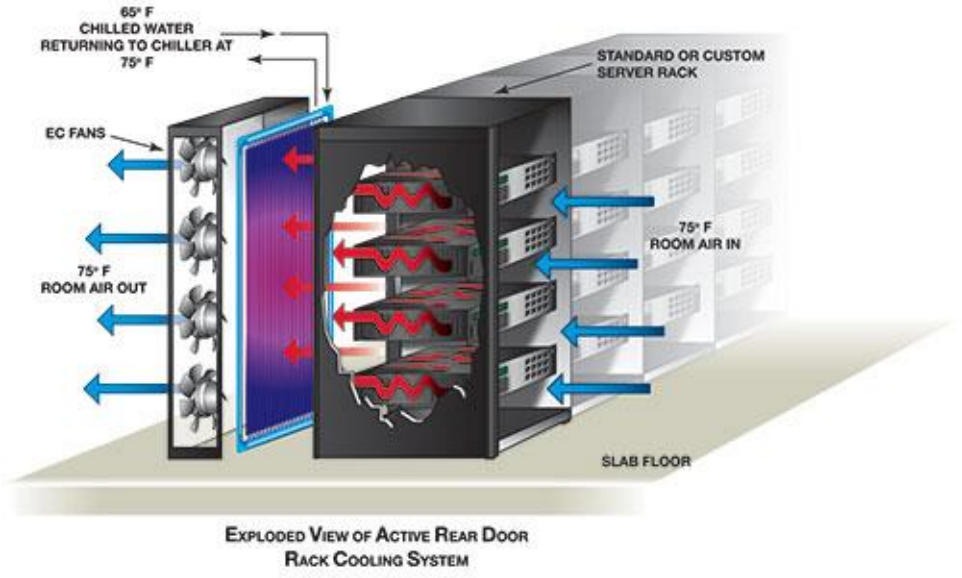
Large traditional single servers would need to be run for long periods without interruption. Availability would have to be very high, making hardware reliability and durability extremely important. Mission-critical enterprise servers would be very fault tolerant and use specialized hardware with low failure rates in order to maximize uptime. Uninterruptible power supplies might be incorporated to insure against power failure. Servers typically include hardware redundancy such as dual power supplies, RAID disk systems, and ECC memory,[8] along with extensive pre-boot memory testing and verification. Critical components might be hot swappable, allowing technicians to replace them on the running server without shutting it down, and to guard against overheating, servers might have more powerful fans or use water cooling. They will often be able to be configured, powered up and down or rebooted remotely, using out-of-band management, typically based on IPMI. Server casings are usually flat and wide, and designed to berack-mounted.

These types of servers are often housed in dedicated data centers. These will normally have very stable power and Internet and increased security. Noise is also less of a concern, but power consumption and heat output can be a serious issue. Server rooms are equipped with air conditioning devices.

5.5 Energy Consumption :

In 2010, data centers (servers, cooling, and other electrical infrastructure) were responsible for 1.1-1.5% of electrical energy consumption worldwide and 1.7-2.2% in the United States.[12] One estimate is that total energy consumption for information and communications technology saves more than 5 times its carbon footprint[13] in the rest of the economy by enabling efficiency.

HP server



Server Cooling System

5.6 Server Responsibility:

In our project the server is responsible for :

1)The server linking between the hardware and the mobile application.

2)The server to control the buying and selling process.

3)The server nomination house with the highest surplus of electricity to carry out the sale and purchase.

4)The server allows the client to determine the price of electricity.

5)The server tells the client in case of a malfunction or if there is an electrical overload the client should alleviate loads.

6)The server keeps the data private customers such as the user name and its password in addition to the Data bass for the network.

5.7 Server Structure :

We have 2 servers :

1)online server :

On cloud.

2)offline server

On Raspberry pi.

The offline server collecting the data on raspberry pi and it uploaded it to the cloud.

5.8 So what is cloud computing ?

To IBM Cloud computing, often referred to as simply “the cloud,” is the delivery of on-demand computing resources—everything from applications to data centers—over the Internet on a pay-for-use basis.

Cloud computing is a type of computing that relies on *sharing computing resources* rather than having local servers or personaldevices to handle applications. Cloud computing is comparable togrid computing, a type of computing where unused processing cycles of all computers in a network are harnesses to solve problems too intensive for any stand-alone machine.

In cloud computing, the word cloud (also phrased as "the cloud") is used as a metaphor for "*the Internet*," so the phrase *cloud computing* means "a type of Internet-based computing," where different services — such as servers, storage and applications —are delivered to an organization's computers and devices through the Internet.

5.9 How Cloud Computing Works?

The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second,

in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive online computer games.

To do this, cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared ITinfrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

5.10 Cloud Computing Standards

The standards for connecting the computer systems and the software needed to make cloud computing work are not fully defined at present time, leaving many companies to define their own cloud computing technologies. Cloud computing systems offered by companies, like IBM's "Blue Cloud" technologies for example, are based on open standards and open source software which link together computers that are used to to deliver Web 2.0capabilities like mash-ups or mobile commerce.

Cloud Computing in the Data Center and for Small Business
Cloud computing has started to obtain mass appeal in corporate data centers as it enables the data center to operate like the Internet through the process of enabling computing resources to be accessed and shared as virtual resources in a secure and scalable manner.

For a small and medium size business (SMB), the benefits of cloud computing is currently driving adoption. In the SMB sector there is often a lack of time and financial resources to purchase, deploy and maintain an infrastructure (e.g. the software, server and storage).

In cloud computing, small businesses can access these resources and expand or shrink services as business needs change. The common pay-as-you-go subscription model is designed to let SMBs easily add or remove services and you typically will only pay for what you do use.

5.11 Cloud Types of service :

Software as a service (SaaS)

Cloud-based applications—or software as a service—run on distant computers "in the cloud" that are owned and operated by others and that connect to users' computers via the Internet and, usually, a web browser.

*The benefits of SaaS

- You can sign up and rapidly start using innovative business apps
- Apps and data are accessible from any connected computer
- No data is lost if your computer breaks, as data is in the cloud
- The service is able to dynamically scale to usage needs

Platform as a service (PaaS)
Platform as a service provides a cloud-based environment with everything required to support the complete lifecycle of building and delivering web-based (cloud) applications—without the cost and complexity of buying and managing the underlying hardware, software, provisioning, and hosting.

*The benefits of PaaS

- Develop applications and get to market faster
- Deploy new web applications to the cloud in minutes
- Reduce complexity with middleware as a service

Infrastructure as a service (IaaS)
Infrastructure as a service provides companies with computing resources including servers, networking, storage, and data center space on a pay-per-use basis.

*The benefits of IaaS

- No need to invest in your own hardware
- Infrastructure scales on demand to support dynamic workloads
- Flexible, innovative services available on demand

Cloud Types :
Public clouds are owned and operated by companies that offer rapid access over a public network to affordable computing resources. With public cloud services, users don't need to purchase hardware, software, or supporting infrastructure, which is owned and managed by providers.

Key aspects of public cloud

- Innovative SaaS business apps for applications ranging from customer resource management (CRM) to transaction management and data analytics
- Flexible, scalable IaaS for storage and compute services on a moment's notice
- Powerful PaaS for cloud-based application development and deployment environments.

A private cloud is infrastructure operated solely for a single organization, whether managed internally or by a third party, and hosted either internally or externally. Private clouds can take advantage of cloud's efficiencies, while providing more control of resources and steering clear of multi-tenancy.

Key aspects of private cloud

- A self-service interface controls services, allowing IT staff to quickly provision, allocate, and deliver on-demand IT resources
- Highly automated management of resource pools for everything from compute capability to storage, analytics, and middleware
- Sophisticated security and governance designed for a company's specific requirements

5.12 Why Cloud computing ?

1. Flexibility

Cloud-based services are ideal for businesses with growing or fluctuating bandwidth demands. If your needs increase it's easy to scale up your cloud capacity, drawing on the service's remote servers. Likewise, if you need to scale down again, the flexibility is baked into the service. This level of agility can give businesses using cloud computing a real advantage over competitors – it's not surprising that CIOs and IT Directors rank 'operational agility' as a top driver for cloud adoption.

2. Disaster recovery

Businesses of all sizes should be investing in robust disaster recovery, but for smaller businesses that lack the required cash and expertise, this is often more an ideal than the reality. Cloud is now helping more organisations buck that trend. According to Aberdeen Group, small

businesses are twice as likely as larger companies to have implemented cloud-based backup and recovery solutions that save time, avoid large up-front investment and roll up third-party expertise as part of the deal.

3. Automatic software updates

The beauty of cloud computing is that the servers are off-premise, out of sight and out of your hair. Suppliers take care of them for you and roll out regular software updates – including security updates – so you don't have to worry about wasting time maintaining the system yourself. Leaving you free to focus on the things that matter, like growing your business.

4. Capital-expenditure Free

Cloud computing cuts out the high cost of hardware. You simply pay as you go and enjoy a subscription-based model that's kind to your cash flow. Add to that the ease of setup and management and suddenly your scary, hairy IT project looks at lot friendlier. It's never been easier to take the first step to cloud adoption.

5. Increased collaboration

When your teams can access, edit and share documents anytime, from anywhere, they're able to do more together, and do it better. Cloud-based workflow and file sharing apps help them make updates in real time and gives them full visibility of their collaborations.

6. Work from anywhere

With cloud computing, if you've got an internet connection you can be at work. And with most serious cloud services offering mobile apps, you're not restricted by which device you've got to hand.

The result? Businesses can offer more flexible working perks to employees so they can enjoy the work-life balance that suits them – without productivity taking a hit. One study reported that 42% of workers would swap a portion of their pay for the ability to telecommute. On average they'd be willing to take a 6% pay cut.

7. Document control

The more employees and partners collaborate on documents, the greater the need for watertight document control. Before the cloud, workers had to send files back and forth as email attachments to be worked on by one user at a time. Sooner or later – usually sooner – you end up with a mess of conflicting file content, formats and titles.

And as even the smallest companies become more global, the scope for complication rises. According to one study, "73% of knowledge workers

collaborate with people in different time zones and regions at least monthly".

When you make the move to cloud computing, all files are stored centrally and everyone sees one version of the truth. Greater visibility means improved collaboration, which ultimately means better work and a healthier bottom line. If you're still relying on the old way, it could be time to try something a little more streamlined.

8. Security
Lost laptops are a billion dollar business problem. And potentially greater than the loss of an expensive piece of kit is the loss of the sensitive data inside it. Cloud computing gives you greater security when this happens. Because your data is stored in the cloud, you can access it no matter what happens to your machine. And you can even remotely wipe data from lost laptops so it doesn't get into the wrong hands.

9. Competitiveness
Wish there was a simple step you could take to become more competitive? Moving to the cloud gives access to enterprise-class technology, for everyone. It also allows smaller businesses to act faster than big, established competitors. Pay-as-you-go service and cloud business applications mean small outfits can run with the big boys, and disrupt the market, while remaining lean and nimble. David now packs a Goliath-sized punch.

10. Environmentally friendly
While the above points spell out the benefits of cloud computing for your business, moving to the cloud isn't an entirely selfish act. The environment gets a little love too. When your cloud needs fluctuate, your server capacity scales up and down to fit. So you only use the energy you need and you don't leave oversized carbon footprints. This is something close to our hearts at Salesforce, where we try our best to create sustainable solutions with minimal environmental impact.

5.13 Why raspberry pi?

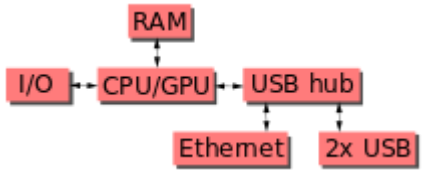
The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer

to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support.

5.14 Hardware :



This block diagram depicts models *A*, *B*, *A*+, and *B*+. Model *A*, *A*+, and *Zero* lack the Ethernet and USB hub components. The Ethernet adapter is connected to an additional USB port. In model *A* and *A*+ the USB port is connected directly to the SoC. On model *B*+ and later models the USB/Ethernet chip contains a five-point USB hub, of which four ports are available, while model *B* only provides two. On the model *Zero*, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port.

1-Processor :

The system on a chip (SoC) used in the first generation Raspberry Pi is somewhat equivalent to the chip used in older smartphones (such as iPhone, 3G, 3GS). The Raspberry Pi is based on the Broadcom BCM2835 SoC,[15] which includes an 700 MHz ARM1176JZF-S processor, VideoCore IV graphics processing unit (GPU), and RAM. It has a Level 1 cache of 16 KB and a Level 2 cache of 128 KB. The Level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

The Raspberry Pi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache.

The Raspberry Pi 3 uses a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache.

2-RAM :

On the older beta model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release model B (and model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with only a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D, possibly also with video decoding (e.g. XBMC).Comparatively the Nokia 701 uses 128 MB for the Broadcom VideoCore IV. For the new model B with 512 MB RAM initially there were new standard memory split files released(arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). But a week or so later the RPF released a new version of start.elf that could read a new entry in config.txt (gpu_mem=xx) and could dynamically assign an amount of RAM (from 16 to 256 MB in 8 MB steps) to the GPU, so the older method of memory splits became obsolete, and a single start.elf worked the same for 256 and 512 MB Raspberry Pis.

The Raspberry Pi 2 and the Raspberry Pi 3 have 1 GB of RAM. The Raspberry Pi Zero has 512 MB of RAM.

3-Networking :

Though the model A and A+ and Zero do not have an 8P8C ("RJ45") Ethernet port, they can be connected to a network using an external user-supplied USB Ethernet or Wi-Fiadapter. On the model B and B+ the Ethernet port is provided by a built-in USB Ethernet adapter using the SMSC LAN9514 chip. The Raspberry Pi 3 is equipped with 2.4 GHz WiFi 802.11n (600 Mbit/s) and Bluetooth 4.1 (24 Mbit/s) in addition to the 10/100 Ethernet port.

4-Video :

The video controller is capable of standard modern TV resolutions, such as HD and Full HD, and higher or lower monitor resolutions and older standard CRT TV resolutions. As shipped (i.e. without custom overclocking) it is capable of the following: 640×350 EGA; 640×480

VGA; 800×600 SVGA; 1024×768 XGA; 1280×720 720p HDTV; 1280×768 WXGAvariant; 1280×800 WXGA variant; 1280×1024 SXGA; 1366×768 WXGA variant; 1400×1050 SXGA+; 1600×1200 UXGA; 1680×1050 WXGA+; 1920×1080 1080p HDTV; 1920×1200 WUXGA.

Higher resolutions, such as, up to 2048×1152, may work or even 3840×2160 at 15 Hz (too low a framerate for convincing video).[33] Note also that allowing the highest resolutions does not imply that the GPU can decode video formats at those; in fact, the Pis are known to not work reliably for H.265 (at those high resolution, at least), commonly used for very high resolutions (most formats, commonly used, up to full HD, do work).

Although the Raspberry Pi 3 does not have H.265 decoding hardware, the CPU, more powerful than its predecessors, is potentially able to decode H.265-encoded videos in software. The Open Source Media Center (OSMC) project said in February 2016:

The new BCM2837 based on 64-bit ARMv8 architecture is backwards compatible with the Raspberry Pi 2 as well as the original. While the new CPU is 64-bit, the Pi retains the original VideoCore IV GPU which has a 32-bit design. It will be a few months before work is done to establish 64-bit pointer interfacing from the kernel and userland on the ARM to the 32-bit GPU. As such, for the time being, we will be offering a single Raspberry Pi image for Raspberry Pi 2 and the new Raspberry Pi 3. Only when 64-bit support is ready, and beneficial to OSMC users, will we offer a separate image. The new quad core CPU will bring smoother GUI performance. There have also been recent improvements to H265 decoding. While not hardware accelerated on the Raspberry Pi, the new CPU will enable more H265 content to be played back on the Raspberry Pi than before.

5.15 Raspberry Pi 3 announced with OSMC support:

The Pi 3's GPU has higher clock frequencies—300 MHz and 400 MHz for different parts—than previous versions' 250 MHz.[36]

The Pis can also generate 576i and 480i composite video signals, as used on old-style (CRT) TV screens through standard connectors—either RCA or 3.5mm phone connector depending on models. The television signal standards supported are PAL-BGHID, PAL-M, PAL-N, NTSC and NTSC-J.

1-Real-time clock :

The Raspberry Pi does not have a built-in real-time clock, and does not "know" the time of day. As alternatives, a program running on the Pi can get the time from a network time server or user input at boot time, thus knowing the time while powered on.

A real-time hardware clock with battery backup, such as the DS1307, which is fully binary coded, may be added (often via the I²C interface).

2-Operating systems :

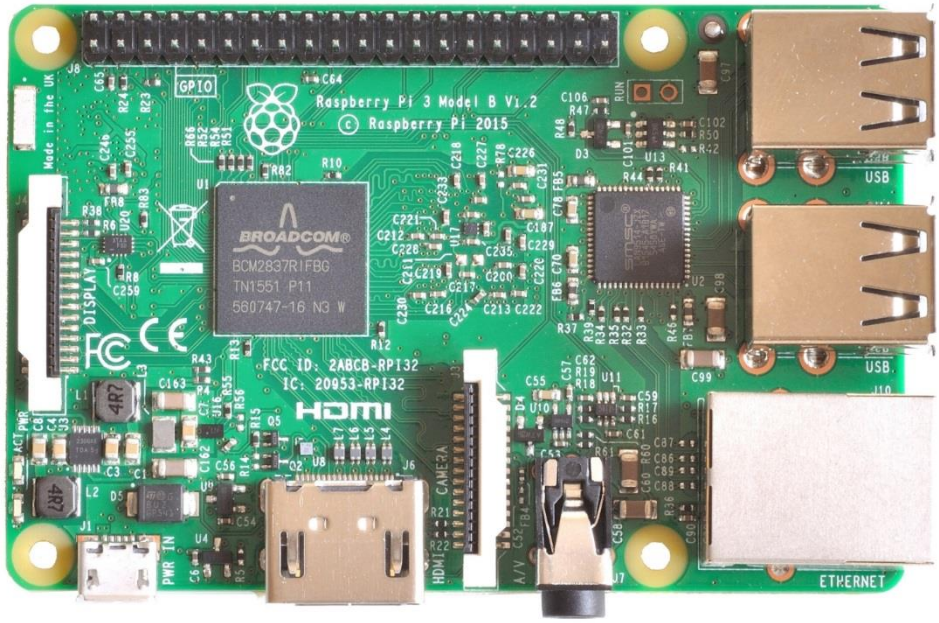
The Raspberry Pi primarily uses Linux-kernel-based operating systems.

The ARM11 chip at the heart of the Pi (first generation models) is based on version 6 of the ARM. The primary supported operating system is Raspbian, although it is compatible with many others. The current release of Ubuntu supports the Raspberry Pi 2, while Ubuntu, and several popular versions of Linux, do not support the older Raspberry Pi 1 that runs on the ARM11. Raspberry Pi 2 can also run the Windows 10 IoT Core operating system , while no version of the Pi can run traditional Windows.The Raspberry Pi 2 currently also supports OpenELEC and RISC OS.

The install manager for the Raspberry Pi is NOOBS. The operating systems included with NOOBS are:

- Arch Linux ARM
- OpenELECOSMC (formerly Raspbmc) and the Kodi open source digital media center
- Pidora (Fedora Remix)
- Puppy Linux
- RISC OS – is the operating system of the first ARM-based computer.
- Raspbian (recommended for Raspberry Pi 1) – is maintained independently of the Foundation; based on the Debian ARM hard-float (armhf) architecture port originally designed for ARMv7 and later processors (with Jazelle RCT/ThumbEE and VFPv3), compiled for the more limited ARMv6 instruction set of the Raspberry Pi 1. A minimum size of 4 GB SD card is required for the Raspbian images provided by the Raspberry Pi Foundation. There is a Pi Store for exchanging programs.
- The Raspbian Server Edition is a stripped version with fewer software packages bundled as compared to the usual desktop computer oriented Raspbian.

- The Wayland display server protocol enables efficient use of the GPU for hardware accelerated GUI drawing functions. On 16 April 2014, a GUI shell for Weston calledMaynard was released.
- PiBang Linux – is derived from Raspbian.
- Raspbian for Robots – is a fork of Raspbian for robotics projects with Lego, Grove, and Arduino.



Raspberry pi model 3

5.16 why firebase ?

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in real time to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Real time Database instance and automatically receive updates with the newest data.

Realtime	**Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive**

	experiences without thinking about networking code.
Offline	Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.
Accessible from Client Devices	The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.

Firebase Realtime Databaseplat_iosplat_androidplat_web
Store and sync data with our NoSQL cloud database. Data is synced across all clients in realtime, and remains available when your app goes offline.

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

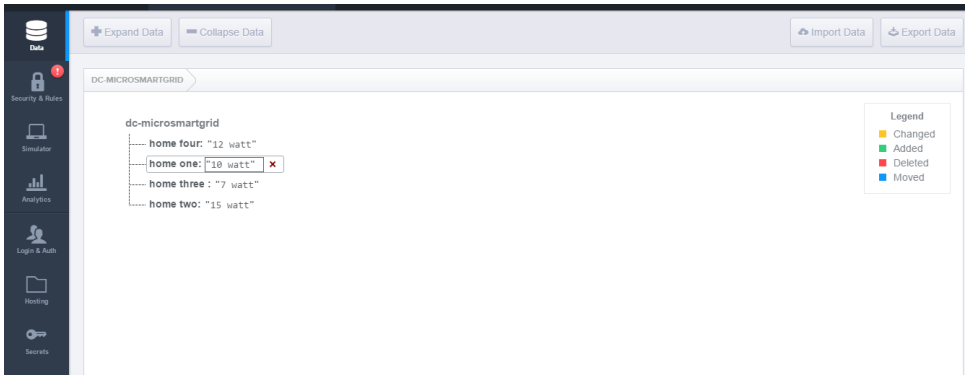
5.17 How does it work?

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes

the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Real time Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Real time Database API is designed to only allow operations that can be executed quickly. This enables you to build a great real time experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.



Our Firebase Application

Our script is written in Python language .

5.18 why python ?

You may have heard that Python is gaining in popularity, but did you know it's now the most popular introductory teaching language in U.S. universities? And it's the fourth most popular language according to an IEEE survey, behind old classics Java, C, and C++? So in celebration of our two new Python courses — Try Python and Flying Through Python — and the launch of our new Python technology Path, I wanted to delve

into why Python is useful to learn, and showcase a few companies who use it.

Python is easy to use, powerful, and versatile, making it a great choice for beginners and experts alike. Python's readability makes it a great first programming language — it allows you to think like a programmer and not waste time understanding the mysterious syntax that other programming languages can require. For instance, look at the following code to print "hello world" in Java and Python.

The Dropbox desktop client is written entirely in Python, which speaks to its cross-platform compatibility. Dropbox has about 400 million users and considering it isn't bundled with any operating system distribution, that's a lot of users downloading and installing Dropbox. In addition to their desktop client, Dropbox's server-side code is in Python as well, making it the majority language used at the company.

Google uses a mix of languages, with C++, Python, and now Go among them. Early on at Google, there was an engineering decision to use "Python where we can, C++ where we must." Python was used for parts that required rapid delivery and maintenance. Then, they used C++ for the parts of the software stack where it was important to have very low latency and/or tight control of memory.

Like Google, Spotify and Netflix use a mix of languages. Spotify uses Java heavily, but uses Python for things like their Web API and their Interactive API console, which lets developers explore endpoints with an easy-to-use interface. Spotify also uses Python for data analytics and other non-customer facing processes, such as a DNS server recovery system, their payment system, and their label content management system. Netflix uses a mix of Java , Scala, and Python, and gives developers autonomy when choosing which language fits the problem best. Where do they use Python most? They heavily use Python and iPython in their real-time analytics group.

If you take a look at these companies, you can see they benefit from Python for its ease of use and because it's great for rapid prototyping and iteration. You can also see that Python can be used for a wide variety of applications, and as you learn the basics of Python, you'll be able to create almost anything you want. Many great developers contribute daily to the Python community by creating Python libraries. These libraries can help you get started so that you don't have to write code to reinvent the wheel. So for example, if you want to do complex image processing,

thePython Imaging Library will help you get started. Want to create games? PyGame is a Python game engine. If data science is your thing, SciPy is the library for you.

- The ability to (eventually) **program on the Web**. Increasingly, the Web is critical to the profession and craft of programming and students should have Web frameworks available when they're ready.
- The ability to program **desktop applications**. While trends are moving more of what we do onto the web, there's nothing like the immediacy of making and running your first local program.
- An eventually **marketable professional skill**. While academic or recreational programming is excellent, the skills we teach should also be usable in a professional context should students choose to use them in that way.
- A **supportive and welcoming community** surrounding the language. Once again this is crucial for those who haven't had exposure to coding from a young age.

5.19 The Script :

The script has three functions

1)Analysis function :
It's a function that it's input is array of Electrical power in the battery every home and it analysis these data and return the house has a more powerful electric batteries in.

2)Buying and selling function :

It's a function tells us that what home you can sell electricity from it and what's the price of electricity.it's input is array of Electrical power in the battery every home and it's out put is the preferred home and the price of electricity.

3)Restart function :

It's a function that makes the Raspberry pi restart after Buying and selling process to make function number one start analysis the new data.it's input is array of Electrical power in the battery every home and it's output is json request .

5.20 Database table :

We used firebase as we said before, the table consists of clo

username	password	power	price
Home one	qwfgbgl145	10 watt	20$
Home two	fsfjkfjsl5	15 watt	17$
Home three	jdsjkjsk56	7 watt	10$
Home four	jkdsjkluo6	12 watt	14$

Database table of DC Micro Smart Grid

5.21 What about security of firebase ?

Firebase provides a flexible, expression-based rules language with JavaScript-like syntax to easily define how your data should be structured and when your data can be read from and written to. Combined with our login service which allows for easy authentication, you can define who has access to what data and keep all of your user's personal information secure. The Security and Firebase Rules live on the Firebase servers and are automatically enforced at all times.

*Authentication:

User identity is an important security concept. Different users have different data, and sometimes they have different capabilities. For example, in a chat program, each message is associated with the user that created it. Users may also be able to delete their own messages, but not messages posted by other users. The first step in securing your app is often identifying your users. This process is called *authentication*.

Firebase provides tools for making authentication easy:

- Integrations with Facebook, GitHub, Google, and Twitter authentication providers
- Email & password login, and account management
- Single-session anonymous login
- Custom login tokens, for integrating with your own authentication server or SSO.

*Authorization:

Identifying your user is only part of security. Once you know who they are, you need a way to control their access to data in your Firebase database.

Firebase has a declarative language for specifying rules that live on the Firebase servers and determine the security of your app

Firebase also support encryption.

5.22 How our server is connecting to our mobile application ?

To make this we

- Create a Firebase account.
- Get an application URL from Firebase.
- Import the client libraries into your app. These are available for iOS, Android, web applications, and REST.
- Call the libraries from your app, referencing the application URL.

Chapter (6): Android

Chapter 6
Android

6.1 Overview:

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast—every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital content.
Android gives you a world-class platform for creating apps and games for Android users everywhere, as well as an open marketplace for distributing to them instantly

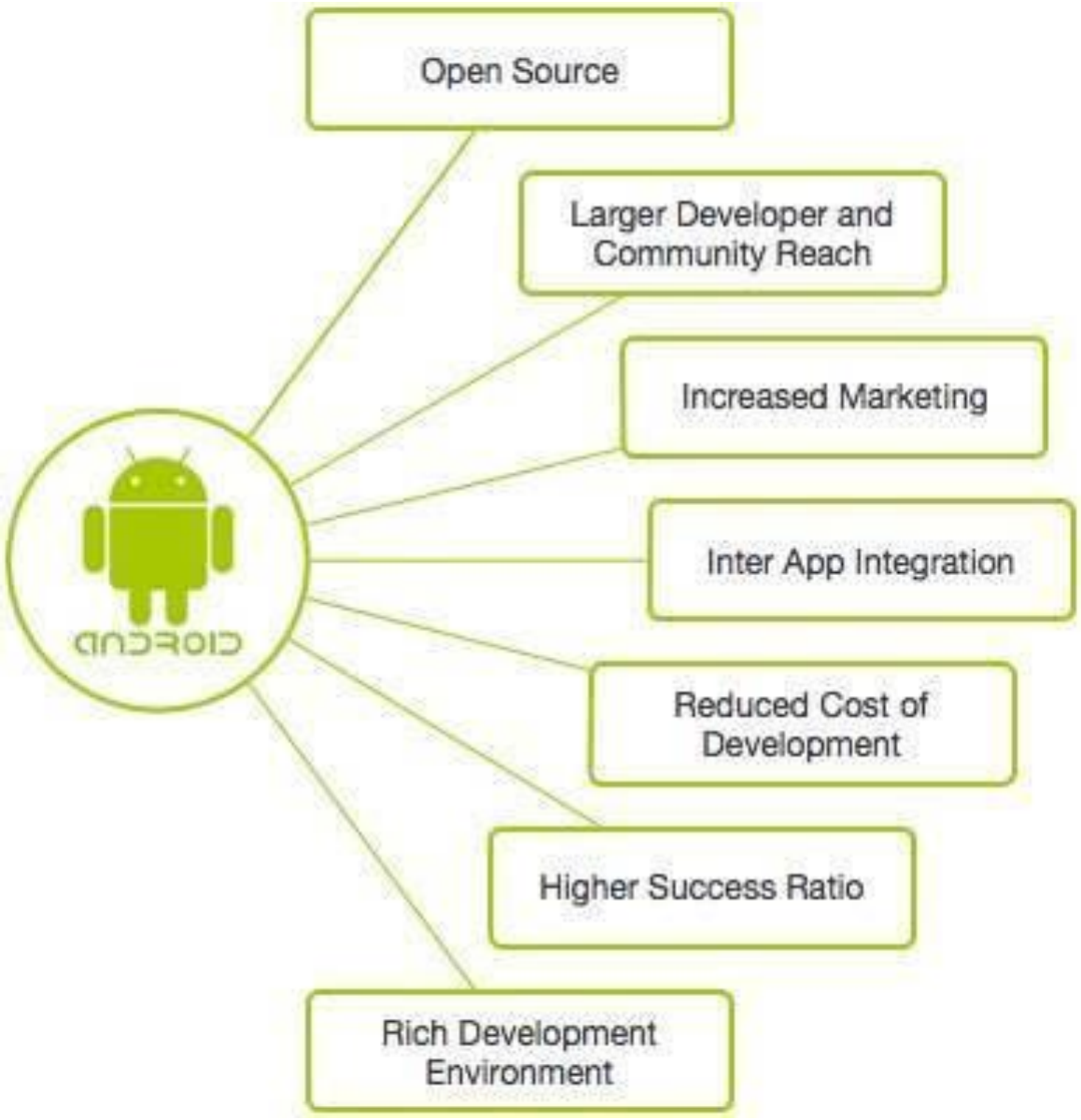
6.2 Introduction:

*What is android?
Operating Systems have developed a lot in last 15 years. Starting from black and white phones to recent smart phones or mini computers, mobile OS has come far away. Especially for smart phones, Mobile OS has greatly evolved from Palm OS in 1996 to Windows pocket PC in 2000 .then to Blackberry OS and Android

One of the most widely used mobile OS these days is ANDROID. Android is a software bunch comprising not only operating system but also middleware and key applications Android Inc. was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005.

Android is a mobile operating system currently developed by Google, based on the Linux and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

*Why android?



6.3 Applications:

Android applications are usually developed in the Java language using the Once developed, Android .Android Software Development Kit applications can be packaged easily and sold out either through a store such as Google Play.

Android offers a unified approach to application development for mobile devices which means developers need only develop for

Android, and their applications should be able to run on different devices .powered by Android

6.4 Features & Specifications:

Android is a powerful operating system, supporting great features. Few of them are listed below:

1-Beautiful UI

Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard] Game controllers and full-size physical keyboards are supported via Bluetooth or USB The response to user input is designed to be immediate and provides a fluid touch interface

2-Memory management

Since Android devices are usually battery-powered, Android is designed to manage processes to keep power consumption at a minimum. When an application is not in use the system suspends its operation so that, while available for immediate use rather than closed, it does not use battery power or CPU resources

3-Open-source community

Android has an active community of developers and enthusiasts who use the Android Open Source Project (AOSP) source code to develop and distribute their own modified versions of the operating system.

Android manages the applications stored in memory automatically: when memory is low, the system will begin invisibly and automatically closing inactive processes, starting with those that have been inactive for longs

4-Store

.Android supporting a large number of applications in Smart Phones

These applications make life more comfortable and advanced for the .users

Android comes with an Android market which is an online software store. .It was developed by Google

It allows Android users to select, and download applications developed .by third party developers and use them

There are around 2.0 lack+ games, application and widgets available on the market for users

5-Connectivity

GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX

6-GCM

Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a

proprietary sync solution.

7-Multi-tasking

User can jump from one task to another and same time various application can run simultaneously.

8- Media support

H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, WAV, JPEG, PNG, GIF, and BMP

9-Resizable widgets

Widgets are resizable, so users can expand them to show more content or shrink them to save space

6.5 History of Android:

The code names of android ranges from A to L currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat and Lollipop. Let's understand the android history in a sequence.

6.6 What is API level?

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Android Versions	Name	API Level	Kernel Version
1.5	Cupcake	3	2.6.27
1.6	Donut	4	2.6.29
2.0/1	Eclair	5-7	2.6.29
2.2 X	Froyo	8	2.6.32
2.3 X	Gingerbread	9, 10	2.6.35
3.0	Honeycomb	11-13	2.6.36
4.0	Ice Cream Sandwich	14, 15	3.0.1
4.1 X	Jelly Bean	16	3.0.31
4.2 X	Jelly Bean	17	3.4.0
4.3	Jelly Bean	18	3.4.39
4.4	Kit Kat	19, 20	3.10
5.0	Lollipop	21, 22	3.16.1
6.0	Marshmallow	23	3.18.10

6.7 Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials.

The Android SDK is composed of modular packages that you can download separately using the Android SDK Manager. For example, when the SDK Tools are updated or a new version of the Android platform is released, you can use the SDK Manager to quickly download them to your environment. Simply follow the procedures described in .Adding Platforms and Packages

There are several different packages available for the Android SDK. The table below describes most of the available packages and where they're .located once you download them

6.8 Available Packages

***SDK Tools**

Contains tools for debugging and testing, plus other utilities that are required to develop an app. If you've just installed the SDK starter package, then you already have the latest version of this package. Make .sure you keep this up to date

***SDK Platform-tools**

Contains platform-dependent tools for developing and debugging your application. These tools support the latest features of the Android platform and are typically updated only when a new platform becomes available. These tools are always backward compatible with older platforms, but you must be sure that you have the latest version of these .tools when you install a new SDK platform

***Documentation**

An offline copy of the latest documentation for the Android .platform APIs

***SDK Platform**

There's one SDK Platform available for each version of Android. It includes an android.jar file with a fully compliant Android library. In order to build an Android app, you must specify an SDK platform as your .build target

***System Images**

Each platform version offers one or more different system images (such as for ARM and x86). The Android emulator requires a system image to operate. You should always test your app on the latest version of Android and using the emulator with the latest system image is a good .way to do so

***Sources for Android SDK**

A copy of the Android platform source code that's useful for .stepping through the code while debugging your app

***Samples for SDK**

A collection of sample apps that demonstrate a variety of the platform APIs. These are a great resource to browse Android app code. The API Demos app in particular provides a huge number of small demos .you should explore

***Google APIs**

An SDK add-on that provides both a platform you can use to develop an app using special Google APIs and a system image for the .emulator so you can test your app using the Google APIs

***Android Support**

A static library you can include in your app sources in order to use powerful APIs that aren't available in the standard platform. For example, the support library contains versions of the Fragment class that's compatible with Android 1.6 and higher (the class was originally introduced in Android 3.0) and the View Pager APIs that allow you to .easily build a side-swappable UI

***Google Play Billing**

Provides the static libraries and samples that allow you to integrate billing services in your app with Google Play.

***Google Play Licensing**

Provides the static libraries and samples that allow you to perform license verification for your app when distributing with Google Play

6.9 App components:

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter your application. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role each one is a unique building block that helps define your application's overall behavior.
There are different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.
Here are the five types of application components:

1-Activities:

An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows.
An application usually consists of multiple activities that are loosely bound to each other. Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity

in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus. The back stack abides to the basic "last in, first out" stack mechanism, so, when the user is done with the current activity and presses the *Back* button, it is popped from the stack (and destroyed) and the previous activity resumes. (The back stack is discussed more in the Tasks and Back Stack document.)

When an activity is stopped because a new activity starts, it is notified of this change in state through the activity's lifecycle callback methods. There are several callback methods that an activity might receive, due to a change in its state—whether the system is creating it, stopping it, resuming it, or destroying it—and each callback provides you the opportunity to perform specific work that's appropriate to that state change. For instance, when stopped, your activity should release any large objects, such as network or database connections. When the activity resumes, you can reacquire the necessary resources and resume actions that were interrupted. These state transitions are all part of the activity lifecycle.
The rest of this document discusses the basics of how to build and use an activity, including a complete discussion of how the activity lifecycle works, so you can properly manage the transition between various activity states

2-Services:
A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform inter process communication (IPC). For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

*A service can essentially take two forms:
a-Started:
A service is "started" when an application component (such as an activity) starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed. Usually, a started service performs a single operation and does not return a result to the caller. For example, it might download or upload a file over the network. When the operation is done, the service should

stop itself.

b-Bound:

A service is "bound" when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

Although this documentation generally discusses these two types of services separately, your service can work both ways—it can be started (to run indefinitely) and also allow binding. It's simply a matter of whether you implement a couple callback methods: onStartCommand() to allow components to start it and onBind() to allow binding.Regardless of whether your application is started, bound, or both, any application component can use the service (even from a separate application), in the same way that any component can use an activity—by starting it with an Intent. However, you can declare the service as private, in the manifest file, and block access from other applications. This is discussed more in the section about Declaring the service in the manifest.

3-Content providers:

Content providers manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security. Content providers are the standard interface that connects data in one process with code running in another process. When you want to access data in a content provider, you use the ContentResolver object in your application's Context to communicate with the provider as a client. The ContentResolver object communicates with the provider object, an instance of a class that implements ContentProvider. The provider object receives data requests from clients, performs the requested action, and returns the results.You don't need to develop your own provider if you don't intend to share your data with other applications. However, you do need your own provider to provide custom search suggestions in your own application. You also need your own provider if you want to copy and paste complex data or files from your application to other applications. Android itself includes content providers that manage data such as audio, video, images, and personal contact information. You can see some of them listed in the reference documentation for the android.provider package. With some restrictions, these providers are accessible to any Android application.

4-Intents and intent filters:
An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use-cases:

- **To start an activity:**

An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to startActivity(). The Intent describes the activity to start and carries any necessary data.

If you want to receive a result from the activity when it finishes, call startActivityForResult(). Your activity receives the result as a separate Intent object in your activity's onActivityResult() callback. For more information, see the Activities guide.

- **To start a service:**

A Service is a component that performs operations in the background without a user interface. You can start a service to perform a one-time operation (such as download a file) by passing an Intent to startService(). The Intent describes the service to start and carries any necessary data.

If the service is designed with a client-server interface, you can bind to the service from another component by passing an Intent to bindService(). For more information, see the Services guide.

- **To deliver a broadcast:**

A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().

5-Processes and Threads:

When an application component starts and the application does not have any other components running, the Android system starts a new Linux process for the application with a single thread of execution. By default, all components of the same application run in the same process and thread (called the "main" thread). If an application component starts and there already exists a process for that application (because another component from the application exists), then the component is started within that process and uses the same thread of execution. However, you can arrange for different components in your application to run in separate processes, and you can create additional threads for any process.

6.10 App Manifest:

Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest file presents essential information about your app to the Android system, information the system must have before it can run any of the app's code. Among other things, the manifest does the following:

- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application — the activities, services, broadcast receivers, and content providers that the application is composed of. It names the classes that implement each of the components and publishes their capabilities (for example, which Intent messages they can handle). These declarations let the Android system know what the components are and under what conditions they can be launched.
- It determines which processes will host application components.
- It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
- It also declares the permissions that others are required to have in order to interact with the application's components.
- It lists the Instrumentation classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.

- It declares the minimum level of the Android API that the application requires.
- It lists the libraries that the application must be linked against.

6.11 Structure of the Manifest File:

The diagram below shows the general structure of the manifest file and every element that it can contain. Each element, along with all of its attributes, is documented in full in a separate file. To view detailed information about any element, click on the element name in the diagram, in the alphabetical list of

```
<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>
```

```
    <service>
        <intent-filter> . . . </intent-filter>
        <meta-data/>
    </service>

    <receiver>
        <intent-filter> . . . </intent-filter>
        <meta-data />
    </receiver>

    <provider>
        <grant-uri-permission />
        <meta-data />
        <path-permission />
    </provider>

    <uses-library />

</application>
```

6.12 User Interface

All user interface elements in an Android app are built using View and ViewGroup objects. A View is an object that draws something on the screen that the user can interact with. A ViewGroup is an object that holds other View (and ViewGroup) objects in order to define the layout of the interface.

Android provides a collection of both View and ViewGroup subclasses that offer you common input controls (such as buttons and text fields) and various layout models (such as a linear or relative layout).

a-User Interface Layout:
The user interface for each component of your app is defined using a hierarchy of View and ViewGroup objects, as shown in figure 1. Each view group is an invisible container that organizes child views, while the child views may be input controls or other widgets that draw some part of the UI. This hierarchy tree can be as simple or complex as you need it to be (but simplicity is best for performance).

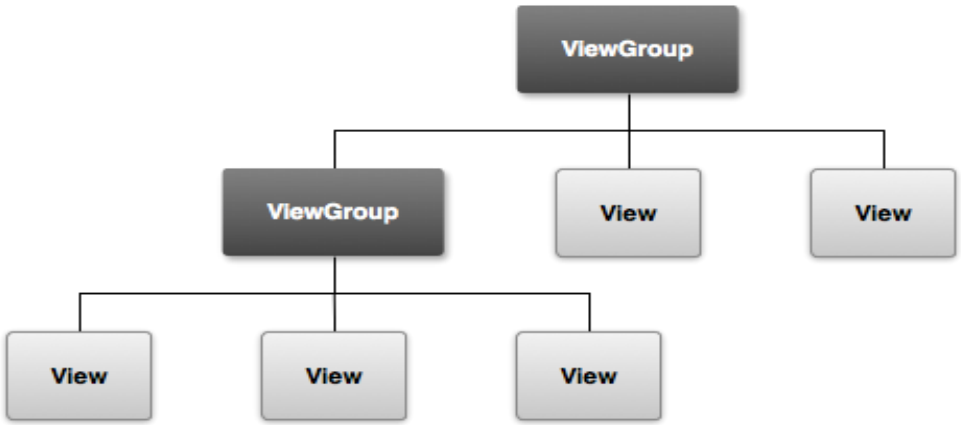


Illustration of a view hierarchy, which defines a UI layout.

To declare your layout, you can instantiate View objects in code and start building a tree, but the easiest and most effective way to define your layout is with an XML file. XML offers a human-readable structure for the layout, similar to HTML.

The name of an XML element for a view is respective to the Android class it represents. So a <TextView> element creates a TextView widget in your UI, and a <LinearLayout> element creates a LinearLayout view group.

For example, a simple vertical layout with a text view and a button looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a Button" />
</LinearLayout>
```

When you load a layout resource in your app, Android initializes each node of the layout into a runtime object you can use to define additional behaviors, query the object state, or modify the layout.

b-User Interface Component:

You don't have to build all of your UI using View and ViewGroup objects. Android provides several app components that offer a standard UI layout for which you simply need to define the content. These UI components each have a unique set of APIs that are described in their respective documents, such as Adding the App Bar, Dialogs, and Status Notifications.

c- Layouts:

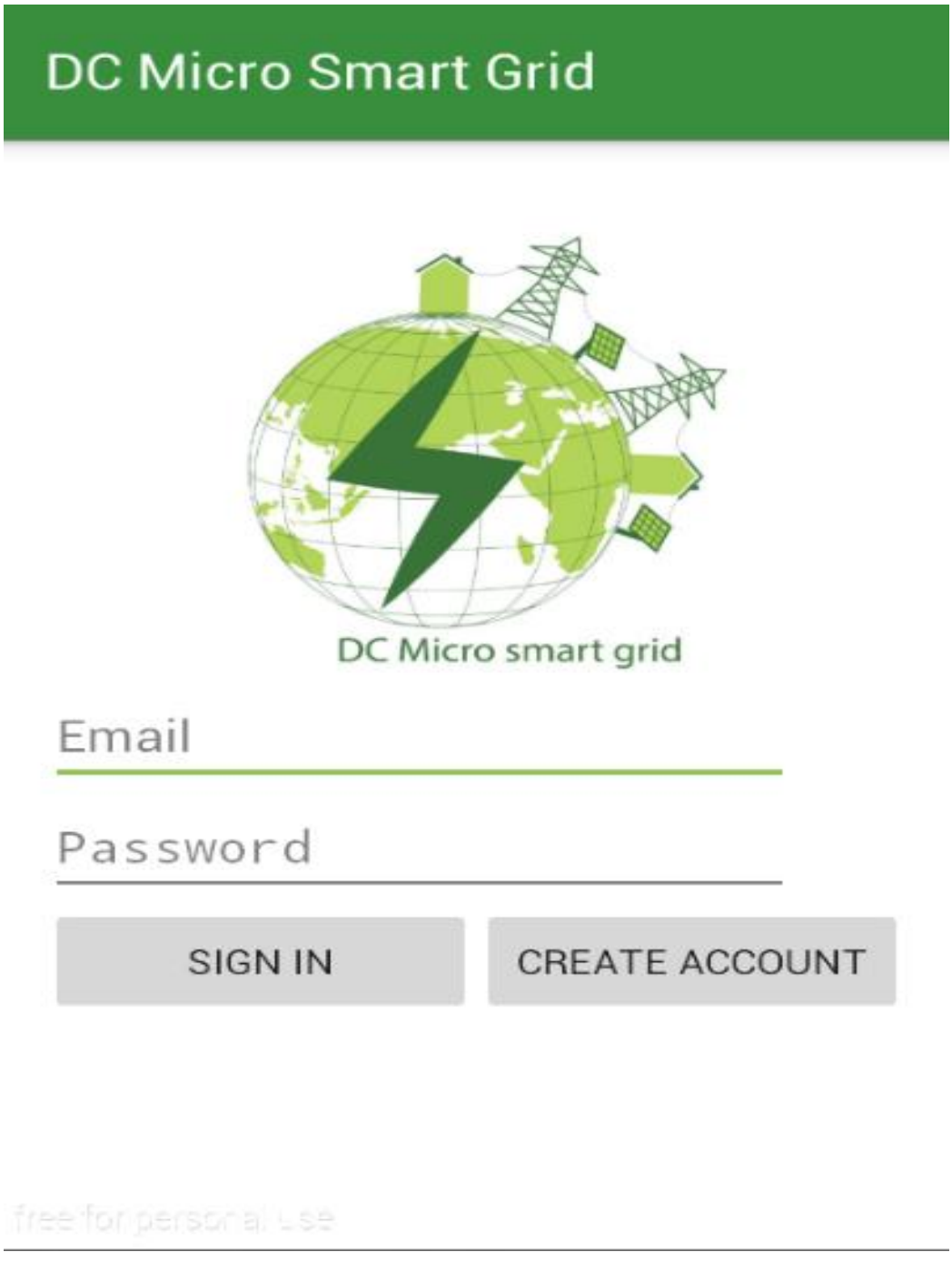
A layout defines the visual structure for a user interface, such as the UI for an activity or app widget. You can declare a layout in two ways:

- Declare UI elements in XML. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- Instantiate layout elements at runtime. Your application can create View and ViewGroup objects (and manipulate their properties) programmatically.

The Android framework gives you the flexibility to use either or both of these methods for declaring and managing your application's UI. For example, you could declare your application's default layouts in XML, including the screen elements that will appear in them and their properties. You could then add code in your application that would modify the state of the screen objects, including those declared in XML, at run time.

The advantage to declaring your UI in XML is that it enables you to better separate the presentation of your application from the code that controls its behavior. Your UI descriptions are external to your application code, which means that you can modify or adapt it without having to modify your source code and recompile. For example, you can create XML layouts for different screen orientations, different device screen sizes, and different languages. Additionally, declaring the layout in XML makes it easier to visualize the structure of your UI, so it's easier to debug problems. As such, this document focuses on teaching you how to declare your layout in XML. If you're interested in instantiating View objects at runtime, refer to the ViewGroup and View class references.

In general, the XML vocabulary for declaring UI elements closely follows the structure and naming of the classes and methods, where element names correspond to class names and attribute names correspond to methods. In fact, the correspondence is often so direct that you can guess what XML attribute corresponds to a class method, or guess what class corresponds to a given XML element. However, note that not all vocabulary is identical. In some cases, there are slight naming differences. For example, the EditText element has a text attribute that corresponds to EditText.setText().



d- Input controls:
Input controls are the interactive components in your app's user interface. Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, checkboxes, zoom buttons, toggle buttons, and many more.
Adding an input control to your UI is as simple as adding an XML element to your XML layout. For example, here's a layout with a text field and button:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage" />
</LinearLayout>
```

Each input control supports a specific set of input events so you can handle events such as when the user enters text or touches a button.

*Common Controls:

Here's a list of some common controls that you can use in your app. Follow the links to learn more about using each one.

DC Micro Smart Grid

HOME

LOADS

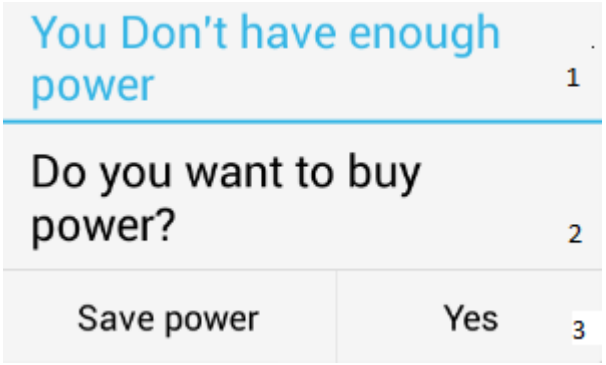
SIGN OUT

Control Type	Description	Related Classes
Button	A push-button that can be pressed, or clicked, by the user to perform an action.	`Button`
Text field	An editable text field. You can use the `AutoCompleteTextView` widget to create a text entry widget that provides auto-complete suggestions	`EditText`, `AutoCompleteTextView`
Checkbox	An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.	`CheckBox`
Radio button	Similar to checkboxes, except that only one option can be selected in the group.	`RadioGroup` `RadioButton`
Toggle button	An on/off button with a light indicator.	`ToggleButton`
Spinner	A drop-down list that allows users to select one value from a set.	`Spinner`
Pickers	A dialog for users to select a single value for a set by using up/down buttons or via a swipe gesture. Use a `DatePicker`code> widget to enter the values for the date (month, day, year) or a `TimePicker` widget to enter the values for a time (hour, minute, AM/PM), which will be formatted automatically for the user's locale.	`DatePicker`, `TimePicke`

e-Dialogs:
A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

*Building an Alert Dialog:

The AlertDialog class allows you to build a variety of dialog designs and is often the only dialog class you'll need. As shown in figure 2, there are three regions of an alert dialog:



1. **Title**

   This is optional and should be used only when the content area is occupied by a detailed message, a list, or custom layout. If you need to state a simple message or question (such as the dialog in figure 1), you don't need a title.

2. **Content area**

   This can display a message, a list, or other custom layout.

3. **Action buttons**

   There should be no more than three action buttons in a dialog.

The AlertDialog.Builder class provides APIs that allow you to create an AlertDialog with these kinds of content, including a custom layout.

*To build an AlertDialog:

```
// 1. Instantiate an AlertDialog.Builder with its constructor
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

// 2. Chain together various setter methods to set the dialog characteristics
builder.setMessage(R.string.dialog_message)
       .setTitle(R.string.dialog_title);

// 3. Get the AlertDialog from create()
AlertDialog dialog = builder.create();
```

f-Notifications:

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the **notification area**. To see the details of the notification, the user opens the **notification drawer**. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

*Design Considerations:
Notifications, as an important part of the Android user interface, have their own design guidelines. The material design changes introduced in Android 5.0 (API level 21) are of particular importance, and you should review the Material Design training for more information. To learn how to design notifications and their interactions, read the Notifications design guide.

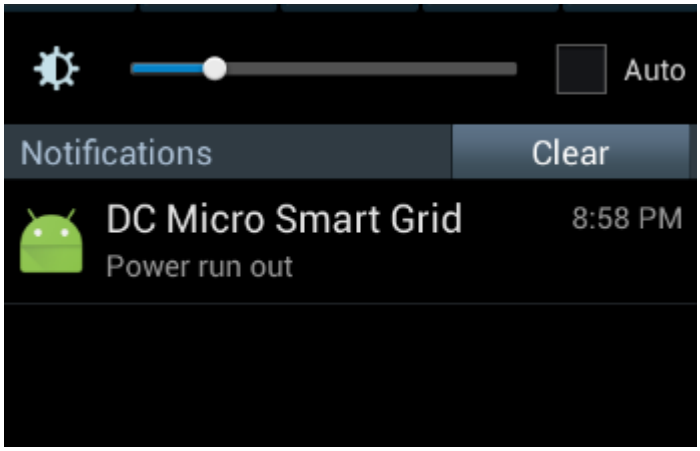
Creating a Notification

You specify the UI information and actions for a notification in a NotificationCompat.Builder object. To create the notification itself, you call NotificationCompat.Builder.build(), which returns a Notification object containing your specifications. To issue the notification, you pass the Notification object to the system by calling NotificationManager.notify().

Required notification contents

A Notification object *must* contain the following:

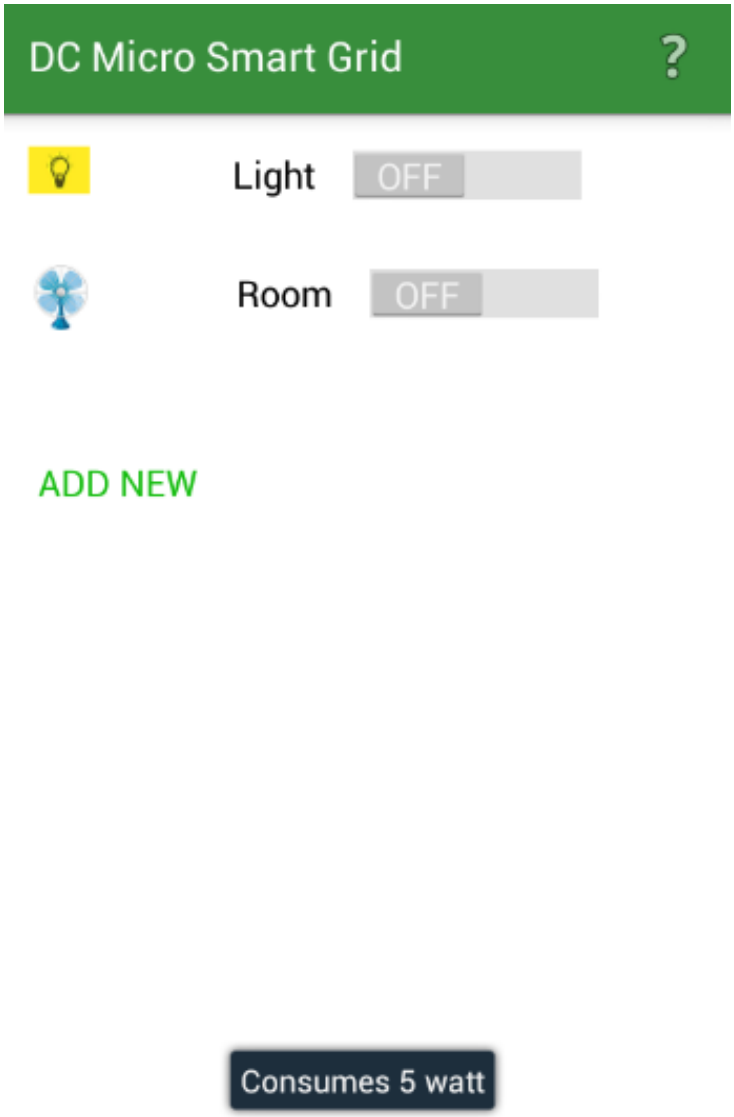
- A small icon, set by setSmallIcon()
- A title, set by setContentTitle()
- Detail text, set by setContentText()



g-Toasts:
A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. For example, navigating away from an email before you send it triggers a "Draft saved" toast to let you know that you can continue editing later. Toasts automatically disappear after a timeout.

*The Basics:
First, instantiate a Toast object with one of the makeText() methods. This method takes three parameters: the application Context, the text message, and the duration for the toast. It returns a properly initialized Toast object. You can display the toast notification with show(), as shown in the following example:

This example demonstrates everything you need for most toast notifications. You should rarely need anything else. You may, however, want to position the toast diferently or even use your own layout instead of a simple text message.



6.13 Firebase:

a powerful platform for building iOS, Android, and web-based apps, offering real-time data storage and synchronization, user authentication, and more.

With Firebase, you can store and sync data to a NoSQL cloud database. The data is stored as JSON, synced to all connected clients in real time, and available when your app goes offline. It offers APIs that enable you to authenticate users with email and password, Facebook, Twitter, GitHub, Google, anonymous auth, or to integrate with existing

authentication system. It also offers hosting for static assets and offers SSL certificates.

Chapter (7): Conclusion

7.1 General Conclusion

A DC Micro Smart Grid consists of interconnected distributed energy resources capable of providing sufficient and continuous energy to a significant portion of internal load demand. With the strong incentives of green and free energy sources and advancement of battery technology, power generation is not just solely relied on coal-fired or gas-fired power plants but everyone can produce their own electricity from these energy sources to power their own loads to certain power level. Many of these sources and applications are of dc nature but today's electricity infrastructure is still based on ac. Therefore there are "unnecessary power processing stages to handle the power generation to the user (i.e. dc to ac and back to dc again)". In addition, the smart use of power electronics converters can help electricity users lower the generation cost and optimize the system efficiency.

In our Project we aimed to achieve these characteristics:

1- Smart: controls the power consumption and offers solutions to save or buy power, all this is dependent on IOT, and by making a mobile application to monitor all data to help consumer in taking descisions.

2- Real time calibrated: provides a continuous updating with the new information about loads, so this helps in specifying resources of power and hence a fast response from customer.
3- Profitable: Selling your excess power and earn money from your grid.
4- Islanded or parallel: It can be operated independently of the utility grid or parallel to it.
5- Renewable: By using solar cells.

There are some Challenges that DC Micro smart grid is suffering from like:

• Increased variability and uncertainty on both demand and supply at distribution levels.
• Bidirectional power flows due to high penetration of distributed resources.
• More frequent outages with high severity caused by extreme weather.
• Increased complexity introduced by voluminous data streams and big data management

• Interoperability between new technologies and with legacy components.
• Secure communications.

7.2 Future Work:

1-There is a control unit that is Responsible for distributing power.

2-The ability to measure the sold and purchased power.

3-Data could be transmitted wirelessly.

4-The ability of mobile application to control the state of loads directly by Bluetooth.

References

- DC Microgrids Scoping Study,
Estimate of Technical and Economic Benefits, March 2015.

- Boonton, Ch1: power measurement Basics.

- Boonton, Ch3: power measurement techniques.

- RS-232, RS-422, RS-485 Serial Communication General Concepts: http://www.ni.com/white-paper/11390/en/
- Basics of the RS-485 Standard: http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/Basics-of-the-RS-485-Standard.aspx

- MAX487 - MAX487 RS485/RS422 Transceivers Technical Data: http://www.futurlec.com/Maxim/MAX487.shtml

- MikroC PRO for PIC Libraries, Hardware Libraries, RS-485 Library.
- Introduction to I²C and SPI protocols:

http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-/protocols

-I2C what is that?
/http://www.i2c-bus.org

-18f4620 Datasheet:

http://ww1.microchip.com/downloads/en/DeviceDoc/39626e.pdf

-16f877A Datasheet:
http://mech.vub.ac.be/teaching/info/mechatronica/PIC16F87XA.pdf

- http://dpnm.postech.ac.kr/cs490/Programming_the_Raspberry_Pi.pdf

- /https://www.raspberrypi.org

- https://en.wikipedia.org/wiki/Raspberry_Pi

https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-
- setup/configuring-i2c

-Comer, Douglas E.; Stevens, David L. (1993).

-Upton, Liz (2 April 2014). Raspberry Pi Foundation(Cambridge).

-Richard A. Henle, Boris W. Kuvshinoff, C. M. Kuvshinoff (1992).

- Adams, James (3 April 2014). "Raspberry Pi Compute Module electrical schematic diagram" (PDF). Raspberry Pi Foundation.

- http://www.ibm.com/us-en/

"Firebase.com Site Info". Alexa Internet.

- https://developer.android.com/index.html

- Smart Grid R&D Program Peer Review Meeting 2014, Smart Grid R&D Program Overview, Dan Ton Acting Deputy Assistant Secretary Power Systems Engineering, June 11, 2014.

- Smart DC Micro-grid for Effective Utilization of Solar Energy D Ravi Prasad, Dr B.Rajesh Kamath, K.R Jagadisha, S.K Girish, International Journal of Scientific & Engineering Research Volume 3, Issue 12, December-2012 1 ISSN 2229-5518.