Sudan University of Science & Technology

College of Engineering

School of mechanical engineering

Temperature Control Over Internet

التحكم في درجة حرارة باستخدام الإنترنت

A project Submitted In Partial Fulfillment for the Requirements of the Degree of B.Sc. (Honor) In Mechanical Engineering

Prepared By:

1. Abdallah Salih Mohammed Salih
2. Ahmed Dia Aldien Hamza Mohammed
3. Mohammed Gasem Ahmed Hussen Ahmed

Supervised By :

Dr. Musaab Hassan Zaroug

September 2015

﴿وَقُلِ الْحَمْدُ لِلَّهِ الَّذِي لَمْ يَتَّخِذْ وَلَدًا وَلَمْ يَكُن لَّهُ شَرِيكٌ فِي الْمُلْكِ وَلَمْ يَكُن لَّهُ وَلِيٌّ مِّنَ الذُّلِّ وَكَبِّرْهُ تَكْبِيرًا﴾

DEDICATION

To those who were very caring, helping and encourage us for advancement and success

Our parents and family

To those who enlighten our way with knowledge since our first steps of education

Our teachers

To those who we knew, spent with them all the moments, happy times and been touch with all meaning of friendship

Our friends and colleagues

ACKNOWLEDGEMENT

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Dr. musaab hassan, for encouragement, guidance, critics and friendship.

My fellow postgraduate students should also be recognized for their support.

My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed.

Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family members.

List of contents

List of Figures

List of Tables

List of abbreviations

PC	Personal Computer
DC	Direct Current
AC	Alternating Current
COM	Communication
GSM	Global System for Mobile
SMS	Short Message Service
DTMF	Dual Tone Multi Frequency
HVAC	Heating, Ventilation and Air Conditioning
LCD	Liquid-Crystal Display
HTTP	Hypertext Transfer Protocol
DAQ	Data Acquisition
RC	Remote Control
HTML	Hypertext Markup Language
MATLAB	Matrix Laboratory
IP	Internet Protocol
PHP	Hypertext Preprocessor
DistrCScentral	Distributed Control System Central
DistrCSnode	Distributed Control System Node
NASA	National Aeronautics and Space Administration
GUI	Graphical User Interface
AVR	A family of microcontroller
IDE	Integrated Development Environment
GPS	Global Positioning System
CAD	Combuter-Aided Design
USB	Universal Serial Bus
GND	Ground

PWM	Pulse-Width Modulation
AREF	Analog Reference
TX	Transmit
RX	Receive
IC	Integrated Circuit
FTDI	Future Technology Device International
SDA	Serial Data
SCL	Serial Clock
ICSP	In-Circuit Serial Programming
VIN	Input Voltage
SRAM	Static Random-Access Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
TTL	Transistor-Transistor Logic
TWI	Two-Wire Interface
SPI	Serial Peripheral Interface
ICSP	In-Circuit Serial Programming
HP	Horse Power
RPM	Round Per Minute
CPU	A Central Processing Unit
PDE	Processing Development Environment
VLW	Visual Language Workshop
HSB	Hue, Saturation and Brightness
RBG	Red, Blue or Green
FAQ	Frequently Asked Questions
RAM	Random Access Memory
WLAN	Wireless Local Area Network

Abstract

The purpose of this project is to develop a system to control the temperature via the internet using client/server technique. This system offers the possibility of remote control in the space which is intended to control through access to the home page online.

This method is an effective way to the fact that internet services are available on a large scale even in the harsh regions, also it can be considered almost a free service.

المستخلص

الغرض من هذا المشروع هو تطوير منظومه للتحكم في درجة الحرارة عن طريق الانترنت باستخدام تقنية الخادم/ المستخدم. توفر هذه المنظومة امكانية التحكم عن بعد في الحيز المراد التحكم فيه وذلك عن طريق الدخول الى الصفحة المخصصة في الانترنت.

هذه الطريقة تعتبر طريقه فعاله لكون خدمات الانترنت متوفرة على نطاق واسع حتى في المناطق النائية, كما يمكن اعتبارها خدمات مجانية.

CHAPTER ONE

INTRODUCTION

CHAPTER ONE

1.1 – Introduction:

The Automatic Temperature Control System was named as a Historic Mechanical Engineering Landmark in 2008.Warren S. Johnson came up with the idea for automatic temperature control while teaching at Norman School in Whitewater, Wisconsin in the 1880's. Originally, janitors would have to enter each classroom to determine if it was too hot or cold and then adjust the dampers in the basement accordingly. Johnson sought a way to end, or at least minimize the classroom interruptions of the janitors and increase the comfort level of the students. The Automatic Temperature Control System would do just that.

Temperature control in manufacturing is a quintessential part of proper product formation. If the temperature slips above or below the ideal range needed for a particular stage in a manufacturing process, the results can be harmful—improperly adhered coatings, a weakened base material, or an overall compromised component—so it becomes increasingly important that the manufacturer not only determine the proper temperature for each stage, but also monitor the temperature inside the machine and receive appropriate feedback .

Temperature controllers in manufacturing operations serve exactly this function, they ensure that a machine is running properly by gauging the temperature at different stages in the process and comparing the data to the programmed temperature specifications. As a result, manufacturers can quickly and easily discover temperature related machine malfunctions, and treat them as necessary.

Controlling over internet technique devotes the internet technologies to link industrial electronics applications over the grid , this is beneficial in providing an obvious and clearly demonstrated interface that’s access remotely from anywhere over the globe , also this approximately a free service its only cost the internet connection fees , since the internet service approximately is available in anywhere even in the harsh terrain and that’s mean the accessibility is high , lastly the updating for the sensory data is happening instantaneously relatively.

1.2 – The problem statement:

A distant device that is need to be controlled and it's user is a roamer that is far away thousands kilometers , the device need to be manipulated instantaneously.

1.3 – Scope:

The area of knowledge of this project consists of two broad areas, computer networks area and automation area, so the scope of this project is client-server model along with microcontroller serial communication technologies.

1.4 – Objectives:

- Continuously monitor and control temperature.
- Operation of the DC motor based on the temperature readings.
- Used DC motor in engineering applications to provide human

comfort.

1.5 – Methodology:

To accomplish this project the work was divided into three phases:

I. **Phase one:** Client-server technique over internet developed by processing software .The client can send the commands through the internet to the computer server which receive the command ,the client application adds a preamble to the command and then encrypt it before transfer it through the internet.

In the other side the server application receives the command, decrypt it and then compare the preamble with a specific value to either accept or drop the command.

II. **Phase two:** A study about Arduino Leonardo, Arduino sketch, Processing sketch, transistor Tip120, diode, DC motor, COM ports and temperature sensor. A test was created for microcontroller by using Arduino software. Then an implementation of the control circuit was created.

III. **Phase three:** Communication between client-server technique and control circuit has been done using Processing sketch and Arduino sketch.

1.6 – Thesis layout:

Chapter 1 is introducing the problem definition, the objectives, the methodology and the scope.

Chapter 2 shows brief review about worldwide researches and experiences of using internet to control and gives and overview of the components used to implement the system.

Chapter 3 Describes the system design and the process.

Chapter 4 Discussion and results.

Chapter 5 Concludes and recommendation.

CHAPTER TWO
LITERATURE REVIEW
AND BACKGROUND

CHAPTER TWO

2.1 - Background:

Tele-operation or controlling a distant applications or devices has implemented using different approaches , using narrow rang communication methods like wireless and Bluetooth which restrict the distant in which the controlled application must be placed , to solve this problem another approaches has been researched like using mobile network and GSM modules to connect the controlled device with the mobile phone , exploiting mobile networks to control the desired device has took many forms , like using SMS to control also using DTMF ,tele-operation also used in space exploration and unmanned application these application mostly used radio communication to control , using internet to control a device is more efficient due it's already established infrastructure and it's low cost compared to the mentioned approaches , it's capability to undertake large multimedia like video stream.

2.2 – Previous Work:

In this study, internet controlled heating ventilating and air-conditioning (HVAC) system has been proposed with programmable sleeping time and automatic operation mode, three steps fan speed unit, adjustable fan angle, a remote control device with a LCD and a web based control unit. A low cost microcontroller to control HVAC system and a PC as an internet server are used.

The system has three different control units (remote control by a hand- device, keypad control mounted on HVAC and web based control).Each control unit has same menu options for users. A data acquisition board provides communications between server and microcontroller. A webcam is used to monitor HVAC parameters and room environment remotely over the internet. The system uses HTTP protocol to control devices. Proposed web based control method remotely works for long distance. As HVAC systems have high voltages, driver circuits with opt-couplers have been used for safety of the system.

I. Colak, et al.in 2008 described the process of controlling the environment of a room via internet using a web –server in the site of concern connected to DAQ board to provide the connection between server and microcontroller , the system has three different control methods a panel mounted in the heating ventilating and air-conditioning (HVAC) system itself , a remote control device (RC) and web-based control , to connect the system with the internet they used HTML web pages to input parameters, that is in the client – side , in the server – side the used apache 2.0.4.8 web – server was installed due to open source coding ,also they used mysql5.0 for database.

They used MATLAB packet program to program the DAQ board and to establish server/client connection, the main board and remote control device was programmed in assembly language . also a web-cam is used to monitor HVAC parameters and room environment remotely over internet the system uses HTTP protocol to control devices this for long distance cases. [1]

A. Ashari in 2010 developed a system for monitoring and controlling distributed applications using the microcontroller and the virtual IP, the general pledge to work with the principle of one each IP address to an address of serial port that is connecting to the circuit microcontroller .

By leveraging IP Aliasing on Linux and make the microcontroller as the equipment being controlled and monitored, then the microcontroller can have a virtual IP, so that controlling and monitoring functions can be performed in a distributed manner. There are three applications have been developed for this system, First, a web based applications using PHP. Second, an application using java for the server namely DistrCScentral (Distributed Control System Central). Third, an application that also using java for on each node is namely DistrCSnode (Distributed Control System Node). The application has IP Virtual that connected to the microcontroller circuit via serial port. A Client computer via browser can be used for monitoring the temperature and provide controlling commands through the server for the node which is monitored and controlled. [2]

J.H.Ahnn in 2007 developed robot for NASA that can be controlled using two kind of communication, wireless communication between mobile Robot and a remote Base Station and serial communication between a remote Base Station and a GUI Application.

The main task of this project has two parts, the former is to program the AVR packet controller module which would enable to wirelessly control the robot, the latter is to program the GUI application which would enable the serially control the Base Station, the GUI application is operated by a joystick with two components, by moving the joystick the GUI application commands movement such as go forward, go backward, turn left and turn right as well as the operation of optional tools. [3]

NİHAN GÖK 7328 The Control of Gas Boiler Card activities via Internet project was initiated on September 2006. The basic idea of this project is implementing a gas boiler card, providing the card's communication with a PC and being able to control the card from another PC. The project mainly consists of three parts: construction of boiler control unit, building the communication between the main unit and a PC, and building the communication between two PC's.

The card activities are under the control of the µC. The µC controls the position of the valve with the voltage given under the control of the pulse width modulation. In order to check the output of the valve position the feedback channel returns this value to the µC. The boiler temperature rises up with the heat coming from the valve and to control the temperature of the boiler, another feedback channel is necessary. The boiler temperature is followed by the temperature sensor .

The µC provides the communication with the outside world as well as controlling the card activities. It communicates with a PC using the RS232 serial communication. The temperature data is received by the PC using this communication channel. The aim of this communication is to give the chance of displaying the data to the user.

The last part of the project is about providing the communication between two PC's. A user from anywhere around the world will be able to follow the change in the temperature of the card by entering to the website prepared. [4]

2.3 – System components:

2.3.1 – Arduino:

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE

(Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

The Arduino hardware and software was designed for artists, designers, hobbyists, hackers, newbies, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smart-phone or your TV! This flexibility combined with the fact that the Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a huge variety of Arduino-based projects.

The microcontroller on the board is programmed using the Arduino programming language and the Arduino development environment. Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, Max-MSP).

The boards can be built by hand or purchased preassembled; the software can be downloaded for free. The hardware reference designs

(CAD files) are available under an open-source license, you are free to adapt them to your needs.

There are many varieties of Arduino boards that can be used for different purposes. but most Arduinos have the majority of these components in common:

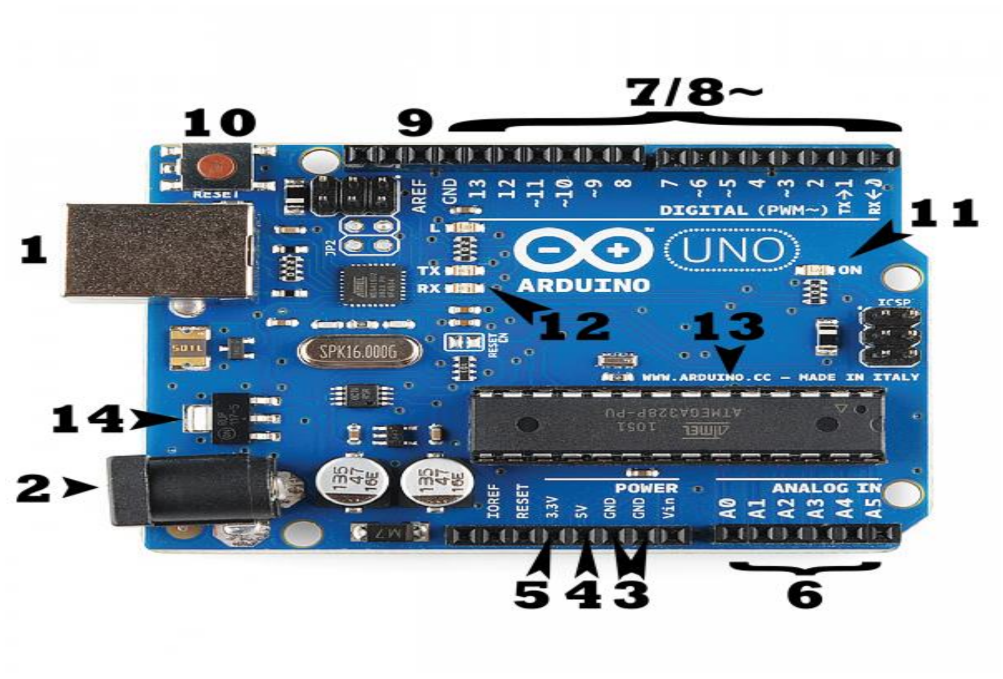


Fig 2-1 Arduino boards

2.3.1.1– Power (USB / Barrel Jack):

Every Arduino board needs a way to be connected to a power source. The Arduino Leonardo can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).

The USB connection is also how you will load code onto your Arduino board.

NOTE: Do not use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

2.3.1.2 – Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF):

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- GND (3): Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- 5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- Analog (6): The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- Digital (7): Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but

for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

- AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

2.3.1.3 – Reset Button:

Just like the original Nintendo, the Arduino has a reset button (10). Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

2.3.1.4 – Power LED Indicator:

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' (11). This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

2.3.1.5 – TX RX LEDs:

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

2.3.1.6 – Main IC:

The black thing with all the metal legs is an IC, or Integrated Circuit (13). Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

2.3.1.7 – Voltage Regulator:

The voltage regulator (14) is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

2.3.1.8 – The Arduino Family:

Arduino makes several different boards, each with different capabilities. In addition, part of being open source hardware means that others can modify and produce derivatives of Arduino boards that provide even more form factors and functionality. Now look at some of the forms of Arduino family:

2.3.1.8.1 – Arduino Uno (R3):

The Uno is a great choice for your first Arduino. It’s got everything you need to get started, and nothing you don’t. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack, a reset button and more. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

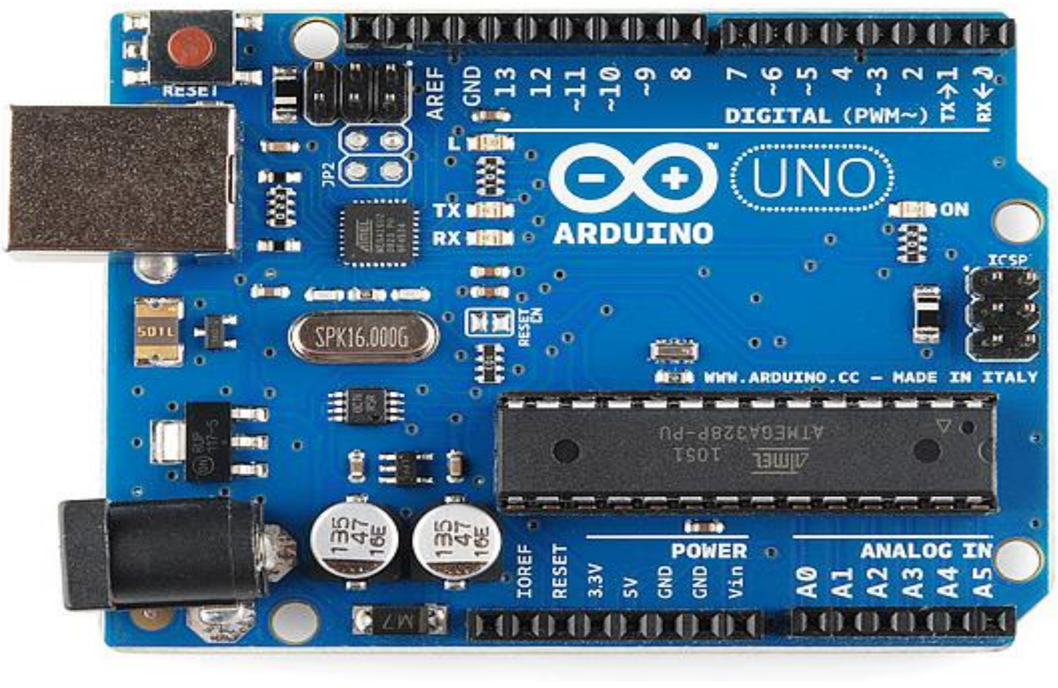


Fig 2-2 Arduino Uno (R3)

2.3.1.8.2 – Lily Pad Arduino:

This is Lily Pad Arduino main board! Lily Pad is a wearable e-textile technology developed by Leah Buechley and cooperatively designed by Leah and Spark Fun. Each Lily Pad was creatively designed with large connecting pads and a flat back to allow them to be sewn into clothing with conductive thread. The Lily Pad also has its own family of input, output, power, and sensor boards that are also built specifically for e-textiles. They're even washable!

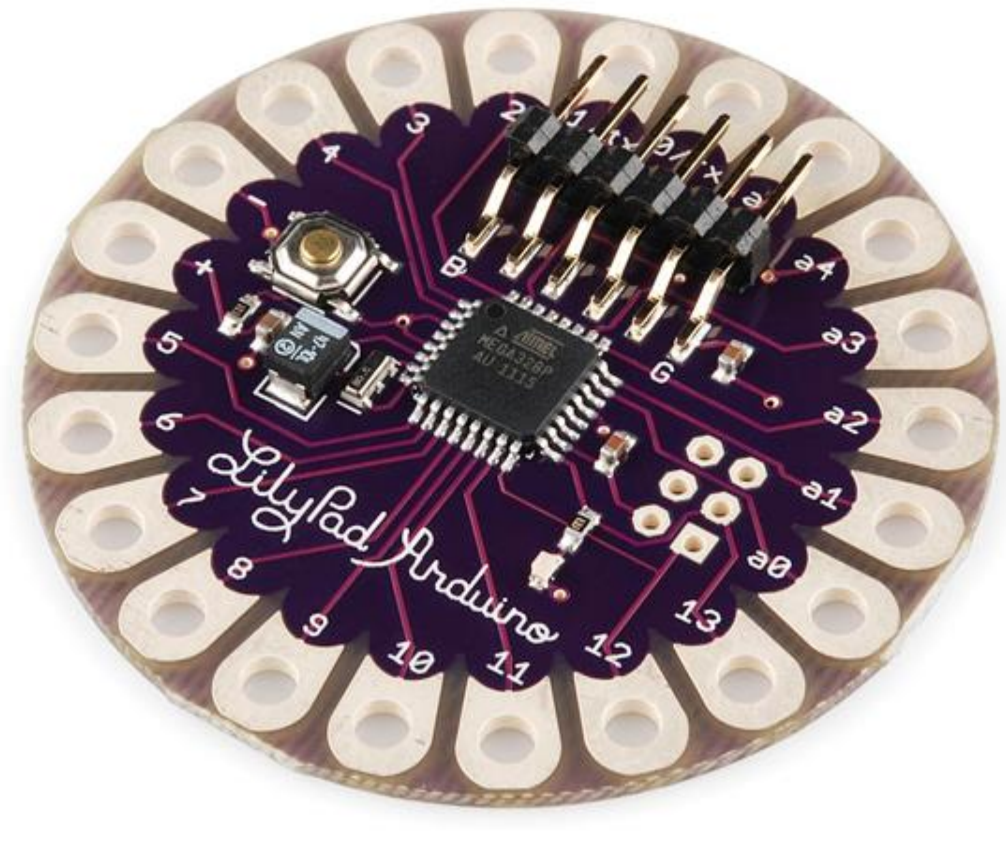


Fig 2-3 Lily Pad Arduino

2.3.1.8.3 – Red Board:

At Spark Fun we use many Arduinos and we're always looking for the simplest, most stable one. Each board is a bit different and no one board has everything we want – so we decided to make our own version that combines all our favorite features.

The Red Board can be programmed over a USB Mini-B cable using the Arduino IDE. It'll work on Windows 8 without having to change your security settings (we used signed drivers, unlike the UNO). It's more stable due to the USB/FTDI chip we used, plus it's completely flat on the back, making it easier to embed in your projects. Just plug in the board, select "Arduino UNO" from the board menu and you're ready to upload code. You can power the Red-Board over USB or through the barrel jack. The on-board power regulator can handle anything from 7 to 15VDC.

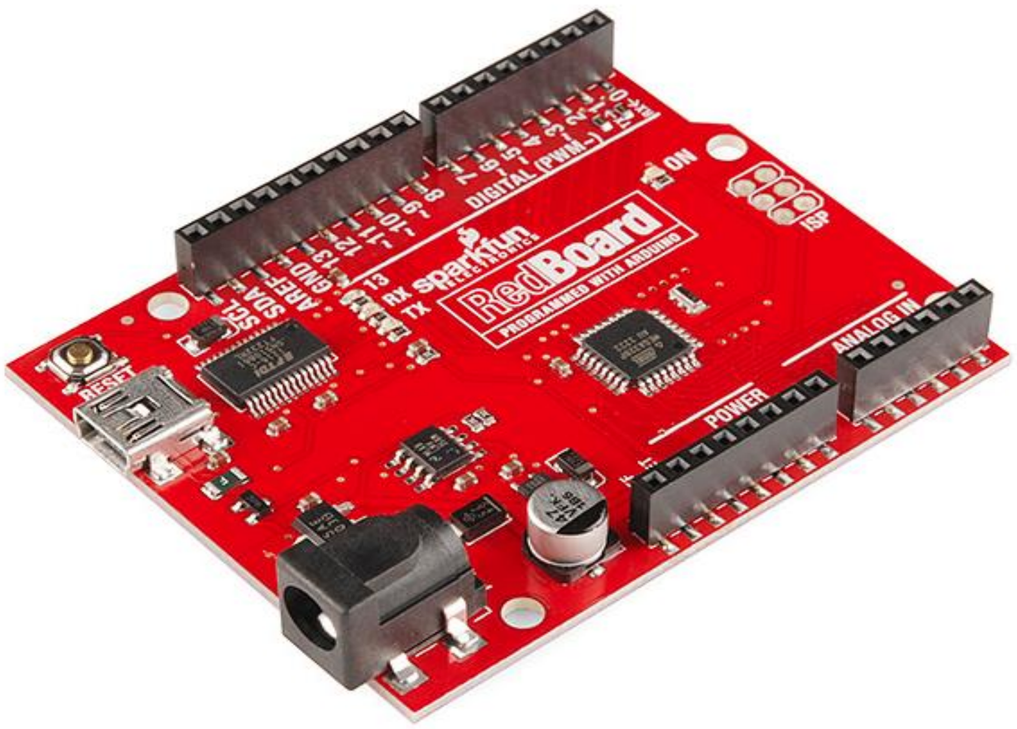


Fig 2-4 Red Board

2.3.1.8.4 – Arduino Mega (R3):

The Arduino Mega is like the UNO’s big brother. It has lots (*54!*) of digital input/output pins (14 can be used as PWM outputs), 16 analog inputs, a USB connection, a power jack, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The large number of pins make this board very handy for projects that require a bunch of digital inputs or outputs (like lots of LEDs or buttons).

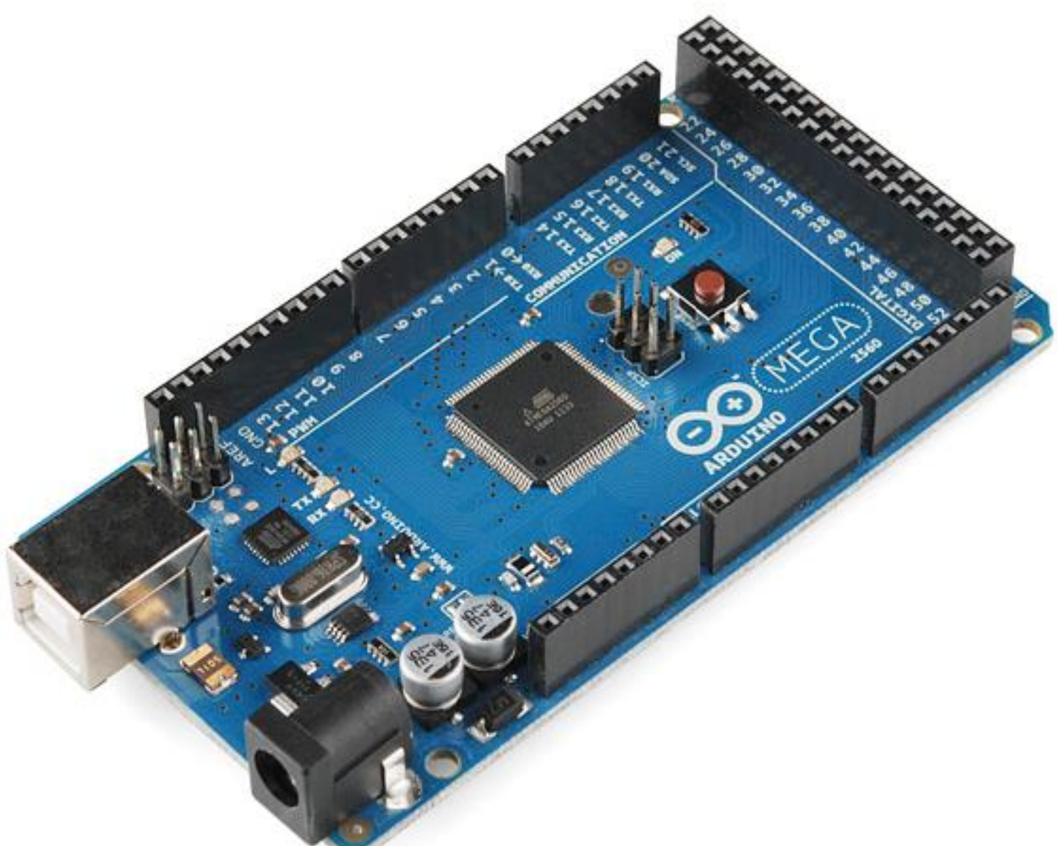


Fig 2-5 Arduino Mega (R3)

2.3.1.8.5 – Arduino Leonardo:

The Leonardo is Arduino’s first development board to use one microcontroller with built-in USB. This means that it can be cheaper and simpler. Also, because the board is handling USB directly, code libraries are available which allow the board to emulate a computer keyboard, mouse, and more!

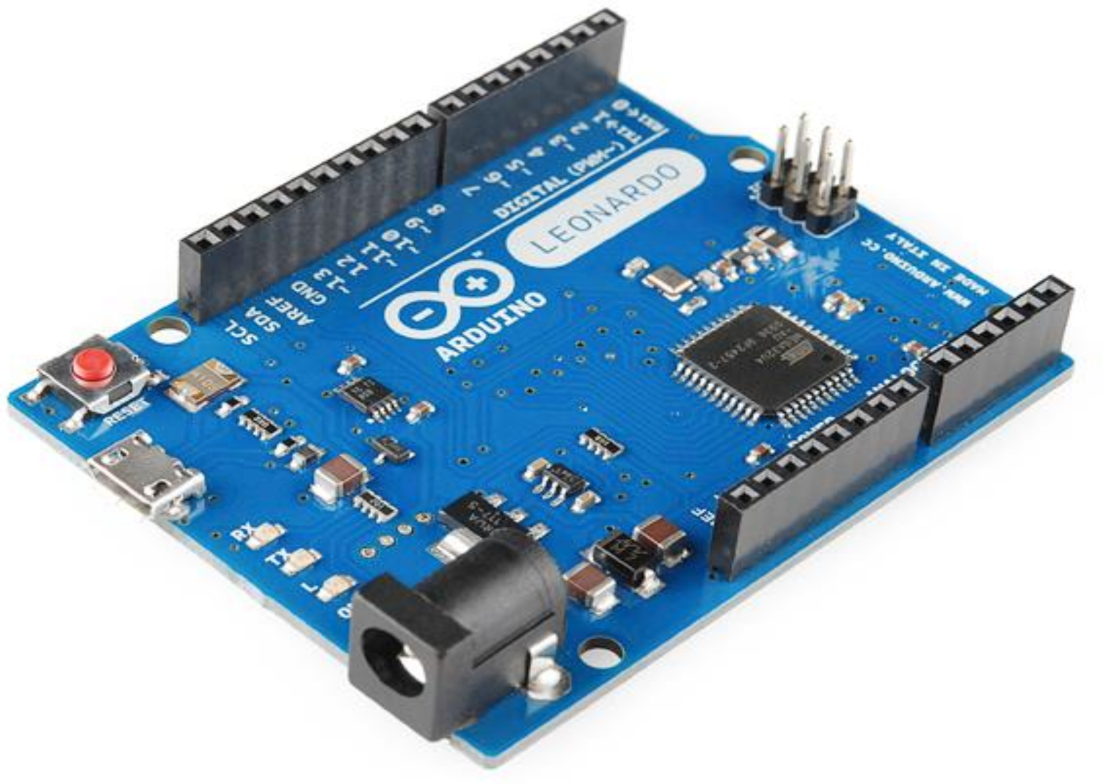


Fig 2-6 Arduino Leonardo

We used Arduino Leonardo in this project because it has the following Features:

- More and Better PWM Pins.
- More Digital Pins.
- More Analog Pins.

- supports USB directly.
- SDA/SCL Pins.
- cheaper than its predecessors.

2.3.1.8.5.1 – Overview:

The Arduino Leonardo is a microcontroller board based on the ATmega32u4 . It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Leonardo differs from all preceding boards in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Leonardo to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port. It also has other implications for the behavior of the board; these are detailed on the getting started page.

2.3.1.8.5.2 – Summary:

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7

Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.3 mm
Weight	20g

2.3.1.8.5.3 – Power:

The Arduino Leonardo can be powered via the micro USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the Power connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts. The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3.3V. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. The voltage at which the i/o pins of the board are operating (i.e. VCC for the board). This is 5V on the Leonardo.

2.3.1.8.5.4 – Memory:

The ATmega32u4 has 32 KB (with 4 KB used for the bootloader). It also has 2.5 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output:

Each of the 20 digital i/o pins on the Leonardo can be used as an input or output, using pin Mode(), digital Write(), and digital Read() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data using the ATmega32U4 hardware serial capability. Note that on the Leonardo, the Serial class refers to USB (CDC) communication; for TTL serial on pins 0 and 1, use the Serial1 class.

- TWI: 2 (SDA) and 3 (SCL). Support TWI communication using the Wire library.
- External Interrupts: 3 (interrupt 0), 2 (interrupt 1), 0 (interrupt 2), 1 (interrupt 3) and 7 (interrupt 4). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt() function for details.
- PWM: 3, 5, 6, 9, 10, 11, and 13. Provide 8-bit PWM output with the analog Write() function.
- SPI: on the ICSP header. These pins support SPI communication using the SPI library. Note that the SPI pins are not connected to any of the digital I/O pins as they are on the Uno, They are only available on the ICSP connector. This means that if you have a shield that uses SPI, but does not have a 6-pin ICSP connector that connects to the Leonardo's 6-pin ICSP header, the shield will not work.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- Analog Inputs: A0-A5, A6 - A11 (on digital pins 4, 6, 8, 9, 10, and 12). The Leonardo has 12 analog inputs, labeled A0 through A11, all of which can also be used as digital i/o. Pins A0-A5 appear in the same locations as on the Uno; inputs A6-A11 are on digital i/o pins 4, 6, 8, 9, 10, and 12 respectively. Each analog input provide 10 bits of resolution (i.e. 1024 different values). By default the analog inputs measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference() function.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analog Reference.

- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega32u4 ports.

2.3.1.8.5.5 – Communication:

The Leonardo has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega32U4 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). The 32U4 also allows for serial (CDC) communication over USB and appears as a virtual com port to software on the computer. The chip also acts as a full speed USB 2.0 device, using standard USB COM drivers. On Windows, a info file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Leonardo's digital pins.

The ATmega32U4 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

The Leonardo appears as a generic keyboard and mouse, and can be programmed to control these input devices using the Keyboard and Mouse classes.

2.3.1.8.5.6 – Programming:

The Leonardo can be programmed with the Arduino software (download). Select "Arduino Leonardo from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega32U4 on the Arduino Leonardo comes pre-burned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the AVR109 protocol.

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

2.3.1.8.5.7 – Automatic (Software) Reset and Boot loader Initiation:

Rather than requiring a physical press of the reset button before an upload, the Leonardo is designed in a way that allows it to be reset by software running on a connected computer. The reset is triggered when the Leonardo's virtual (CDC) serial / COM port is opened at 1200 baud and then closed. When this happens, the processor will reset, breaking the USB connection to the computer (meaning that the virtual serial / COM port will disappear). After the processor resets, the bootloader starts, remaining active for about 8 seconds. The bootloader can also be initiated by pressing the reset button on the Leonardo. Note that when the board first powers up, it will jump straight to the user sketch, if present, rather than initiating the bootloader.

Because of the way the Leonardo handles reset it's best to let the Arduino software try to initiate the reset before uploading, especially if you are in the habit of pressing the reset button before uploading on other boards. If the software can't reset the board you can always start the bootloader by pressing the reset button on the board.

2.3.1.8.5.8 – USB Overcurrent Protection:

The Leonardo has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

2.3.1.8.5.9 – Physical Characteristics:

The maximum length and width of the Leonardo PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins. [6]

2.3.2 – Diode:

A diode is a specialized electronic component with two electrodes called the anode and the cathode. Most diodes are made with semiconductor materials such as silicon, germanium, or selenium. Some diodes are comprised of metal electrodes in a chamber evacuated or filled with a pure elemental gas at low pressure. Diodes can be used as rectifiers, signal limiters, voltage regulators, switches, signal modulators, signal mixers, signal demodulators, and oscillators.

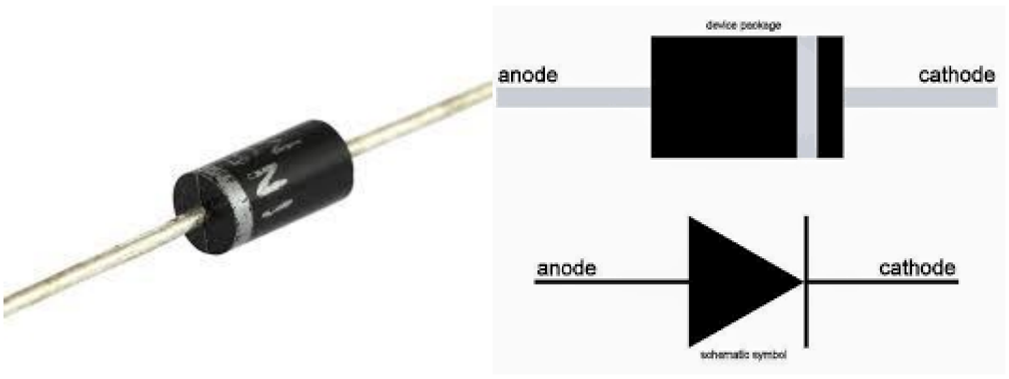


Fig 2-7 Diode

The fundamental property of a diode is its tendency to conduct electric current in only one direction. When the cathode is negatively charged relative to the anode at a voltage greater than a certain minimum called forward break-over, then current flows through the diode. If the cathode is positive with respect to the anode, is at the same voltage as the anode, or is negative by an amount less than the forward break-over voltage, then the diode does not conduct current. This is a simplistic view, but is true for diodes operating as rectifiers, switches, and limiters. The forward break-over voltage is approximately six tenths of a volt (0.6 V) for silicon devices, 0.3 V for germanium devices, and 1 V for selenium devices. [8]

2.3.3 – Resistor:

A resistor is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. Resistors can also be used to provide a specific voltage for an active device such as a transistor. Resistors can be fabricated in a variety of ways. The most common type in electronic devices and systems is the carbon-composition resistor. Fine granulated carbon (graphite) is mixed with clay and hardened. The resistance depends on the proportion of carbon to clay; the higher this ratio, the lower the resistance.

2.3.4 – DC Motor:

Almost every mechanical movement that we see around us is accomplished by an electric motor. Electric machines are means of converting energy. Motors take electrical energy and produce mechanical energy. Electric motors are used to power hundreds of devices we use in everyday life.

Motors come in various sizes; huge motors that can take loads of 1000’s of HP are typically used in the industry for examples electric trains, heavy metal rolling mills and hoists. And small motors applications include motors that’s used in automobile, robots, hand power tools and food blenders.

Electric motors are broadly classified into two different categories: DC – Direct Current and AC – Alternating Current, within these categories are numerous types, each offering unique abilities that suit them well for specific applications. In most cases, regardless of type , electric motors consist of a stator (stationary field) and rotor (the rotating field or armature) and operate through the interaction of magnetic flux and electric current to produce rotational speed and torque. DC motors are distinguished by their ability to operate from direct current. DC motors consist of one set of coils, called armature winding, inside another set of coils or a set of permanent magnets, called the stator. Applying a voltage to the coils produces torque in the armature, resulting in motion , show the internal structure of DC motor in figure:

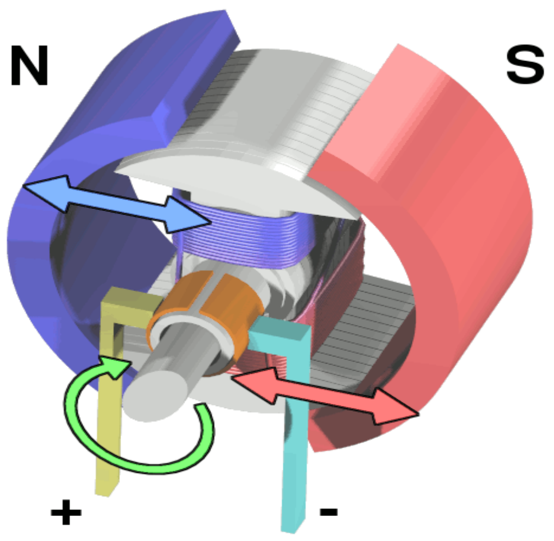


Fig 2-8 DC Motor

2.3.4.1 – Stator:

- The stator is the stationary outside part of a motor.
- The stator of permanent magnet dc motor is composed of two or more permanent magnet pole pieces.
- The magnetic field can alternatively be created by an electromagnet.
- In this case, a DC coil (field winding) is wound around a magnetic material that forms part of the stator.

2.3.4.2 – Rotor:

- The rotor is the inner part which rotates.
- The rotor is composed of windings (called armature windings) which are connected to the external circuit through a mechanical commutator.

- Both stator and rotor are made of ferromagnetic material. The two are separated by air-gap.

2.3.4.3 – Winding:

A winding is made up of series or parallel connection of coil.

- Armature winding – The winding through which the voltage is applied or induced.
- Field winding – The winding through which a current is passed to produce flux (for the electromagnet).
- Windings are usually made of copper.

DC motor also differentiated in basis of the operating voltages used, the it starts from 5v through 12, 24, 48, 96 and etc. voltages, the voltages is also important because of the relation between the RPM (Round Per Minute) and the voltage is proportional. [6]

2.3.5 – Transistor:

Transistors are active components and are found everywhere in electronic circuits. They are used as amplifiers and switching devices. As amplifiers, they are used in high and low frequency stages, oscillators, modulators, detectors and in any circuit needing to perform a function. In digital circuits they are used as switches.

There is a large number of manufacturers around the world who produce semiconductors (transistors are members of this family of components), so there are literally thousands of different types. There are low, medium and high power transistors, for working with high and low frequencies, for working with very high current and/or high voltages. Materials most commonly used are silicon, gallium-arsenide, and germanium, into which impurities have been introduced by a process called "doping." In *n* -type semiconductors the impurities or dopants result in an excess of electrons, or negative charges; in *p* -type semiconductors the dopants lead to a deficiency of electrons and therefore an excess of positive charge carriers or "holes".

The *n*-*p*-*n* junction transistor consists of two *n* -type semiconductors (called the emitter and collector) separated by a thin layer of *p* -type semiconductor (called the base). The transistor action is such that if the electric potentials on the segments are properly determined, a small current between the base and emitter connections results in a large current between the emitter and collector connections, thus producing current amplification. Some circuits are designed to use the transistor as a switching device; current in the base-emitter junction creates a low-resistance path between the collector and emitter. The *p*-*n*-*p* junction transistor, consisting of a thin layer of *n* -type semiconductor lying between two *p* -type semiconductors, works in the same manner, except that all polarities are reversed. [9]

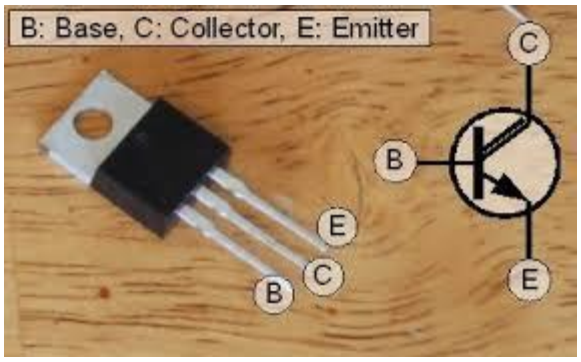


Fig 2-9 Transistor (Tip120)

2.3.6 – LM35:

LM35 is a precision IC temperature sensor with its output proportional to the temperature (in ^{o}C). The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. With LM35**,** temperature can be measured more accurately than with a thermistor. It also possess low self-heating and does not cause more than 0.1 ^{o}C temperature rise in still air.

The operating temperature range is from -55°C to 150°C. The output voltage varies by 10mV in response to every ^{o}C rise/fall in ambient temperature, *i.e.*, its scale factor is 0.01V/ ^{o}C. [10]

2.3.6.1 – Features:

- Calibrated directly in ˚ Celsius (Centigrade)
- Linear + 10.0 mV/˚C scale factor
- 0.5˚C accuracy guaranteeable (at +25˚C)
- Rated for full −55˚ to +150˚C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08˚C in still air
- Nonlinearity only ±1⁄4˚C typical
- Low impedance output, 0.1 Ω for 1 mA load

2.3.6.2 – Pin Description:

Table 2-1: LM35 Describtion

Pin No	**Function**	**Name**
1	Supply voltage; 5V (+35V to -2V)	Vcc
2	Output voltage (+6V to -1V)	Output
3	Ground (0V)	Ground

2.3.6.3 – Pin Diagram:

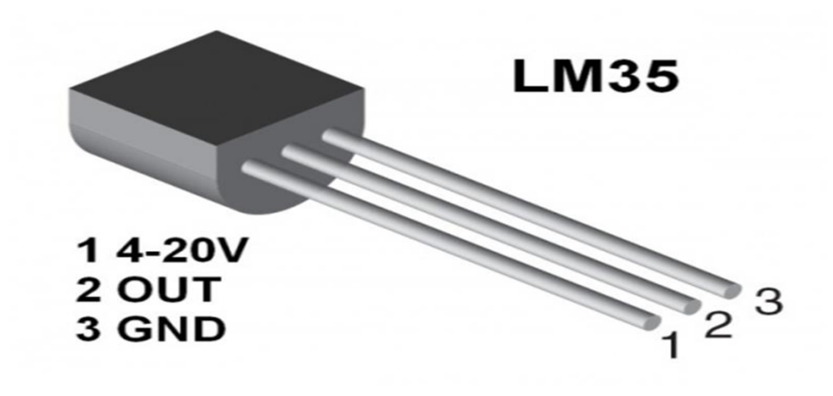


Fig 2-9 Transistor (Tip120)

2.3.7 – Arduino Sketch:

2.3.7.1 – Arduino Software (IDE):

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

2.3.7.2 – Writing Sketches:

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

- ☒ Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

- ☒ Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save

Saves your sketch.

Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools and Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

2.3.7.2.1 – File:

- New :

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

- Open :

Allows to load a sketch file browsing through the computer drives and folders.

- Open Recent :

Provides a short list of the most recent sketches, ready to be opened.

- Sketchbook :

Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

- Examples :

Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.

- Close :

  Closes the instance of the Arduino Software from which it is clicked.

- Save :

  Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.

- Save as... :

  Allows to save the current sketch with a different name.

- Page Setup :

  It shows the Page Setup window for printing.

- Print :

  Sends the current sketch to the printer according to the settings defined in Page Setup.

- Preferences :

  Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.

- Quit :

  Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

2.3.7.2.2 – Edit:

- Undo/Redo :

  Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.

- Cut :

  Removes the selected text from the editor and places it into the clipboard.

- Copy :

Duplicates the selected text in the editor and places it into the clipboard.

- Copy for Forum :

Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

- Copy as HTML :

Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

- Paste :

Puts the contents of the clipboard at the cursor position, in the editor.

- Select All :

Selects and highlights the whole content of the editor.

- Comment/Uncomment :

Puts or removes the // comment marker at the beginning of each selected line.

- Increase/Decrease Indent :

Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.

- Find :

Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

- Find Next :

Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.

- Find Previous :

Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

2.3.7.2.3 – Sketch:

- Verify/Compile :

Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

- Upload :

Compiles and loads the binary file onto the configured board through the configured Port.

- Upload Using Programmer :

This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch.

- Export Compiled Binary :

Saves a .hex file that may be kept as archive or sent to the board using other tools.

- Show Sketch Folder :

Opens the current sketch folder.

- Include Library :

Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

- Add File... :

Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side of the toolbar.

2.3.7.2.4 – Tools:

- Auto Format :

This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

- Archive Sketch :

Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

- Fix Encoding & Reload :

Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

- Serial Monitor :

Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

- Board :

Select the board that you're using.

- Port :

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

- Programmer :

For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

- Burn Bootloader :

The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board.

2.3.7.2.5 – Help:

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- Find in Reference :

This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

2.3.7.3 – Sketchbook:

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

2.3.7.4 – Tabs, Multiple Files, and Compilation:

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

2.3.7.5 – Uploading:

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or/dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx ,/dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

2.3.7.6 – Libraries:

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #includestatements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch.

2.3.7.7 – Third-Party Hardware:

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "Arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

2.3.7.8 – Serial Monitor:

Displays serial data being sent from the Arduino or Genuino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to Serial begin in your sketch. Note that on Windows, Mac or Linux, the Arduino or Genuino board will reset (rerun your sketch execution to the beginning) when you connect with the serial monitor.

You can also talk to the board from Processing, Flash, MaxMSP, etc.

2.3.7.9 – Preferences:

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

2.3.7.10 – Language Support:

Since version 1.0.1 , the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

2.3.7.11 – Boards:

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. [7]

2.3.8 – Processing Sketch:

2.3.8.1 – Overview:

Processing is a simple programming environment that was created to make it easier to develop visually oriented applications with an emphasis on animation and providing users with instant feedback through interaction. The developers wanted a means to “sketch” ideas in code. As its capabilities have expanded over the past decade, Processing has come to be used for more advanced production-level work in addition to its sketching role. Originally built as a domain-specific extension to Java targeted towards artists and designers, Processing has evolved into a full-blown design and prototyping tool used for large-scale installation work, motion graphics, and complex data visualization.

Processing is based on Java, but because program elements in Processing are fairly simple, you can learn to use it even if you don't know any Java. If you're familiar with Java, it's best to forget that Processing has anything to do with Java for a while, until you get the hang of how the API works.

The Processing Development Environment (PDE) makes it easy to write Processing programs. Programs are written in the Text Editor and started by pressing the Run button. In Processing, a computer program is called a sketch. Sketches are stored in the Sketchbook, which is a folder on your computer.

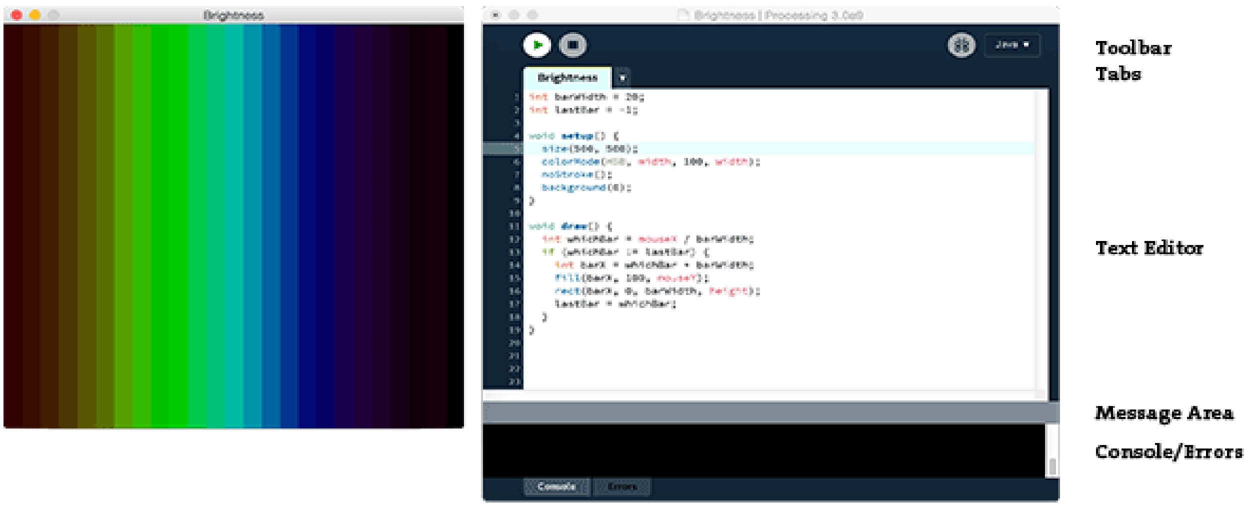
Sketches can draw two- and three-dimensional graphics. The default renderer is for drawing two-dimensional graphics. The P3D renderer makes it possible to draw three-dimensional graphics, which includes controlling the camera, lighting, and materials. The P2D renderer is a fast, but less accurate renderer for drawing two-dimensional graphics. Both the P2D and P3D renderers are accelerated if your computer has an OpenGL compatible graphics card.

The capabilities of Processing are extended with Libraries and Tools. Libraries make it possible for sketches to do things beyond the core Processing code. There are hundreds of libraries contributed by the Processing community that can be added to your sketches to enable new things like playing sounds, doing computer vision, and working with advanced 3D geometry. Tools extend the PDE to help make creating sketches easier by providing interfaces for tasks like selecting colors.

Processing has different programming modes to make it possible to deploy sketches on different platforms and program in different ways. The Java mode is the default. Other programming modes may be downloaded by selecting "Add Mode..." from the menu in the upper-right corner of the PDE.

2.3.8.2 – Processing Development Environment (PDE):

The Processing Development Environment (PDE) consists of a simple text editor for writing code, a message area, a text console, tabs for managing files, a toolbar with buttons for common actions, and a series of menus. The menus options change from mode to mode. The default Java mode is documented here.



```
int barWidth = 20;
int lastBar = -1;

void setup() {
  size(500, 500);
  colorMode(HSB, width, 100, width);
  noStroke();
  background(0);
}

void draw() {
  int whichBar = mouseX / barWidth;
  if (whichBar != lastBar) {
    int barX = whichBar * barWidth;
    fill(barX, 100, mouseY);
    rect(barX, 0, barWidth, height);
    lastBar = whichBar;
  }
}
```

Fig 2-11 Processing Sketch

Programs written using Processing are called sketches. These sketches are written in the text editor. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by Processing sketches including complete error messages and text output from sketches with the print() and print ln() functions. (Note that the console works well for occasional messages, but is not intended for high-speed, real-time output.)

The buttons on the toolbar can run and stop programs,:

- <u>Run :</u>

Runs the sketch. In Java mode, it compiles the code and opens a new display window.

- Stop :

  Terminates a running sketch.

Additional commands are found within the six menus: File, Edit, Sketch, Debug, Tools and Help. The menus are context sensitive which means only those items relevant to the work currently being carried out are available.

2.3.8.2.1 – File:

- New :

  Creates a new sketch in a new window, named as the current date is the format "sketch_ YYMMDDa".
- Open... :

  Open a sketch in a new window.
- Sketchbook :

  Open a sketch from the sketchbook folder.
- Open Recent :

  Select a sketch to open from the list of recently closed sketches.
- Sketchbook... :

  Open a new window to show the list of sketches in the sketchbook.
- Examples... :

  Open a new window to show the list of the examples.
- Close :

  Close the sketch in the front most window. If this is the last sketch that's open, you will be prompted whether you would like to quit. To avoid the prompt, use Quit instead of Close when you want to exit the application.

- Save :

  Saves the open sketch in its current state.
- Save as... :

  Saves the currently open sketch, with the option of giving it a different name. Does not replace the previous version of the sketch.

- Export :

  Exports a Java application as an executable file and opens the folder containing the exported files.
- Page Setup :

  Define page settings for printing.
- Print (Ctrl+P) :

  Prints the code inside the text editor.
- Preferences :

  Change some of the ways Processing works.
- Quit :

  Exits the Processing Environment and closes all Processing windows.

2.3.8.2.2 – Edit:

- Undo :

  Reverses the last command or the last entry typed. Cancel the Undo command by choosing Edit » Redo.
- Redo :

  Reverses the action of the last Undo command. This option is only available if there has already been an Undo action.
- Cut :

  Removes and copies selected text to the clipboard (an off-screen text buffer).

- Copy :

  Copies selected text to the clipboard.

- Copy as HTML :

  Formats code as HTML in the same way it appears in the Processing environment and copies it to the clipboard so it can be pasted somewhere else.

- Paste :

  Inserts the contents of the clipboard at the location of the cursor, and replaces any selected text.

- Select All :

  Selects all of the text in the file which is currently open in the text editor.

- Auto Format :

  Attempts to format the code into a more human-readable layout. Auto Format was previously called Beautify.

- Comment/Uncomment :

  Comments the selected text. If the selected text is already commented, it uncomments it.

- Increase Indent :

  Indents the selected text two spaces.

- Decrease Indent :

  If the text is indented, removes two spaces from the indent.

- Find... :

  Finds an occurrence of a text string within the file open in the text editor and gives the option to replace it with a different text.

- Find Next :

  Finds the next occurrence of a text string within the file open in the text editor.

- Find Previous :

Finds the previous occurrence of a text string within the file open in the text editor.

- Use Selection for Find :

Sets the currently selected text as the item to find with Find Next and Find Previous.

2.3.8.2.3 – Sketch:

- Run :

Runs the code (compiles the code, opens the display window, and runs the sketch inside)

- Present :

Runs the code in the center of the screen with a solid-color background. Click the "stop" button in the lower left to exit the presentation or press the Escape key. Change the background color in the Preferences.

- Tweak :

Runs the code in a way where some color and variable values can be changed while the code is running. The sketch needs to be saved before it can be run as a sketch to Tweak.

- Stop :

If the code is running, stops the execution. Programs written without using the draw() function are stopped automatically after they draw.

- Import Library :

Adds the necessary import statements to the top of the current sketch. For example, selecting Sketch » Import Library » pdf adds the statement "import processing.pdf.*;" to the top of the file. These import statements are necessary for using Libraries. Select Add Libraries... to open the Library Manager to browse and install new libraries.

- Show Sketch Folder :

  Opens the folder for the current sketch.

- Add File... :

  Opens a file navigator window. Select an image, font, or other media files to add it to the sketch's "data" folder.

2.3.8.2.4 – Debug:

- Enable Debugger :
- Continue :
- Step :
- Step Into :
- Step Out :
- Toggle Breakpoint :

2.3.8.2.5 – Tools:

- Create Font... :

  Converts fonts into the Processing font format (VLW) and adds to the current sketch. Opens a dialog box that gives options for setting the font, its size, if it is anti-aliased (smooth), and which characters to be generated. The amount of memory required for the font is determined by the size selected and the number of characters selected through the "Characters..." menu; Processing fonts are textures, so larger fonts require more image data. Fonts can also be created in the code with the create Font() function.

- Color Selector... :

  Interface for selecting colors. For each color, the HSB, RBG, and Hex values are shown. The Hex value can be copied into the clipboard with the Copy button.

- Archive Sketch :

  Archives a copy of the current sketch in .zip format. The archive is placed in the same folder as the sketch.

- Install "processing-java" :
- Movie Maker :

  Creates a QuickTime movie from a sequence of images. Options include setting the size, frame rate, and compression, as well as an audio file.

- Add Tool... :

  Opens the Tool Manager to browse and install new Tools.

2.3.8.2.6 – Help:

- Environment :

  Opens the reference for the Processing Development Environment (this page) in the default web browser.

- Reference :

  Opens the reference in the default web browser. Includes references for the language, programming environment, and core libraries.

- Find in Reference :

  Select an element of the Processing language in the text editor and select Find in Reference to open that page in the default web browser.

- Libraries Reference :
- Tools Reference :

- <u>Getting Started :</u>

  Opens the online Getting Started tutorial in the default browser.
- <u>Troubleshooting :</u>

  Opens the online Troubleshooting wiki page in the default browser.
- <u>Frequently Asked Questions :</u>

  Opens the online FAQ wiki page in the default browser.
- <u>The Processing Foundation :</u>

  Opens the Foundation website in the default browser.

- <u>Visit Processing.org :</u>

  Opens Processing website in the default browser.

2.3.8.3 – Preferences:

The Processing Development Environment (PDE) is highly configurable. The most common preferences can be modified in the Preferences window, located in the File menu on Windows and Linux and in the Processing menu on Mac Os X. The full list of preferences are stored in the "preferences.txt" file. This file can be opened and edited directly only when Processing is not running. You can find the location of this file on your computer by reading the bottom-left corner of the Preferences window.

- <u>Sketchbook location :</u>

  Any folder can be used as the Sketchbook. Input a new location or select "Browse" to set the folder you want to use.
- <u>Language :</u>

  Select the language to use for the menus. Processing needs to be restarted after making a new selection.

- Editor and Console font :

  Select a different font to use for text in the Editor and Console.

☒ Note: the selected font should match the language used in the Text Editor.

- Editor font size :

  Sets the font size of the code in the text editor.

- Console font size :

  Sets the font size of the text in the console.

- Background color when Presenting :

  Defined the background color used when a sketch is run with Present

- Use smooth text in editor window :

  By default, the text in the editor is aliased. When checked, the editor switches to an anti-aliased (smoothed) font. Restart Processing after making this change.

- Enable complex text input :

  Enables the Text Editor to display non-Latin fonts such as Japanese. Processing needs to be restarted after making this selection.

- Continuously check for errors and Show warnings :

  Turn on and off the features that continuously check for and report potential code errors.

- Code completion with Ctrl-space :
- Suggest import statements :
- Increase maximum available memory :

  Allocates more RAM to Processing sketches when they run. Sketches that use media files (images, audio, etc.) sometimes require more RAM. Increase the amount of RAM if a sketch is throwing Out of Memory Errors.

- Delete previous folder on export :

When checked (default behavior), Processing deletes the complete export folder before re-creating it and adding the new media.

- Check for updates on startup :

When checked (default behavior), you'll be informed of new Processing software releases as they become available through a small dialog box that opens as Processing starts.

- Run sketches on display :

If more than one monitor is attached, select the monitor on which to display the sketch.

2.3.8.4 – Sketches and Sketchbook:

All Processing projects are called sketches. Each sketch has its own folder. The main file for each sketch has the same name as the folder and is found inside. For example, if the sketch is named "Sketch_123", the folder for the sketch will be called "Sketch_123" and the main file will be called "Sketch_123.pde". The PDE file extension is an acronym for the Processing Development Environment.

Processing sketches can be stored anywhere on your computer, but by default they are stored in the sketchbook, which will be in different places on your computer or network depending if you use PC, Mac, or Linux and how the preferences are set. To locate this folder, select the "Preferences" option from the File menu (or from the "Processing" menu on the Mac) and look for the "Sketchbook location."

A sketch folder sometimes contains other folders for media files and other code. When a font or image is added to a sketch by selecting

"Add File..." from the Sketch menu, a "data" folder is created. Files may also be added to your Processing sketch by dragging them into the text editor. Image and sound files dragged into the application window will automatically be added to the current sketch's "data" folder. All images, fonts, sounds, and other data files loaded in the sketch must be in this folder.

2.3.8.5 – Renderers:

Processing has six built-in screen renderers. The default renderer is for drawing two-dimensional shapes. P2D is a faster, but less accurate renderer for drawing two-dimensional shapes. P3D is for three-dimensional geometry; it can also control the camera, lighting, and materials. The P2D and P3D renderers are accelerated if your computer has an OpenGL compatible graphics card. Each of these three renderers has a "_2X" version for rendering on high definition screens. To render for a high def screen, use the JAVA2D_2X, P2D_2X, or P3D_2X depending on the type of sketch. The smooth() function affects the amount of antialiasing for each renderer. Check the reference for smooth() for more information.

The renderer used for each sketch is specified through the size() function. If a renderer is not explicitly defined in size(), it uses the default renderer as shown in the following program:

```
void setup() {

  size(200, 200);

}
```

```
void draw() {

  background(204);

  line(width/2, height/2, mouseX, mouseY);

}
```

To change the renderer, add a third parameter to size().

For example:

```
void setup() {

  size(200, 200, P2D);

}

void draw() {

  background(204);

  line(width/2, height/2, mouseX, mouseY);

}
```

A large effort has been made to make Processing code behave similarly across the different renderers, but there are currently some inconsistencies that are explained in the reference.

For more information, see the size() reference entry.

2.3.8.6 – Tabs, Multiple Files, and Classes:

It can be inconvenient to write a long program within a single file. When Processing sketches grow to hundreds or thousands of lines, breaking them into modular units helps manage the different parts. Processing manages files with the Sketchbook and each sketch can have multiple files that are managed with tabs.

The arrow button to the right of the tabs in the Processing Development Environment is used to manage these files. Click this button to reveal options to create a new tab, rename the current tab, and delete the current tab. Tabs are intended for more advanced users, and for this reason, the menu that controls the tabs is intentionally made less prominent.

2.3.8.7 – Programming Modes:

Processing has different programming modes to make it possible to deploy sketches on different platforms and program in different ways. The current default programming mode is Java mode. Other programming modes such as Android Mode and Python are added by selecting "Add Mode..." from the menu in the upper-right corner of the PDE.

2.3.8.8 – Java Mode:

This mode makes it possible to write short programs to draw to the screen, but also enables complex Java programs as well. It can be used simply by beginners, but it scales to professional Java software development. Sketches written in this mode can be exported as Java Applications to run on Linux, Mac OS X, and Windows operating systems.

2.3.8.9 – Adding Libraries, Tools, and Modes:

Processing 3.0 includes a set of features to make it easier to install, update, and remove Libraries, Tools, Modes, and Examples.

Add a contributed library by selecting "Add Library..." from the "Import Library..." submenu within the Sketch menu. This opens the Library Manager. Next, select a library and then click on Install to download it.

Add a contributed tool by selecting "Add Tool..." from the Tools menu, then select a Tool to download from the Tool Manager.

Add contributed modes by selecting "Add Mode..." from the Mode menu in the upper-right corner of the PDE, then select a Mode to install.

Add contributed Examples by first opening the "Examples..." submenu from the File menu. Click on the Add Examples button to open the Examples Manager. Next, select an examples package and select Install to download.[11]

CHAPTER THREE

SYSTEM DESIGN

CHAPTER THREE

3.1 –Introduction:

In this chapter we will describe the practical implementation of the project which include the system block diagram procedure of the circuit and codes which were written in the Arduino sketch and processing sketch.

3.2 –System Block Diagram:

The block diagram of the system is generally described in figure 3-1 the client is the end of this system who is a roaming individual. The client sends the commands to the server computer through Internet. The server received the commands and transfer it through serial communication to the Arduino.

Arduino get the temperature value from LM35 sensor, After that Arduino transfer signal to dc motor. The operation of Arduino works in two directions (sending and receiving).

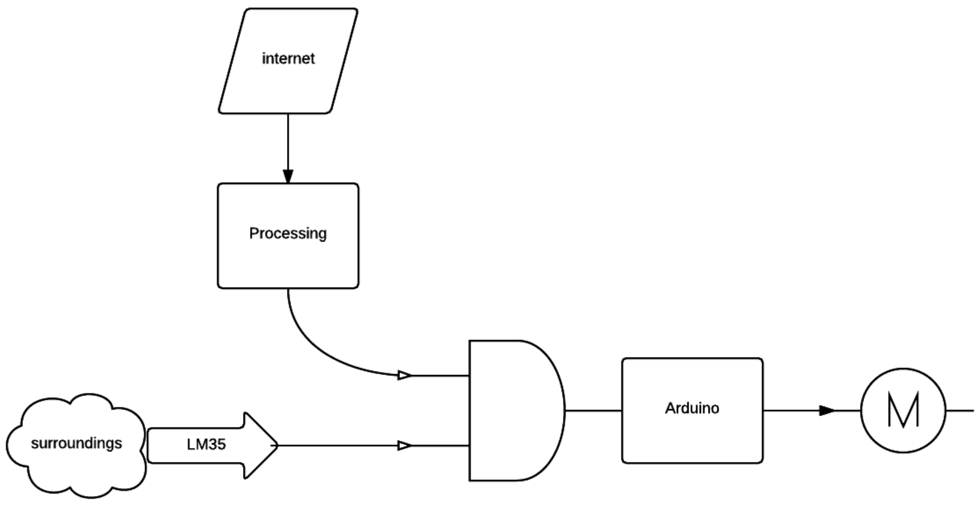


Fig 3-1 System Block Diagram

3.3 –Procedure:

The diode end was connected with the Tip120 collector and the other end of the diode (silver part) connected with battery (+ev) and motor (+ev). We also connected motor(-ev) with Tip120 collector, and connected the battery (-ev) plus Arduino (PWM) ground with (Tip120) Emitter , and connecting (Tip120) Base with pin13 in Arduino which is a digital pin.

We also connect the three pins as follow:

- +5V to be connected with +5V Arduino.

- Sensor's ground with Arduino's ground.

- Sensor's output with Arduino A0 output which is (Analog pin).

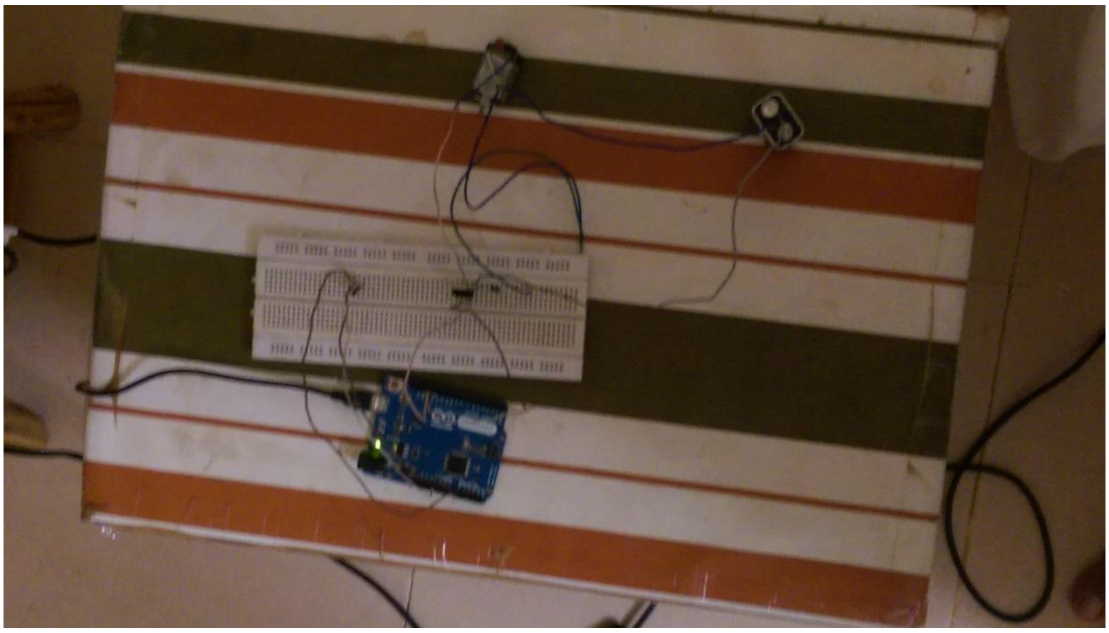


Fig 3-2 The System Implementation

When we opened the web page(Temperaturecontrol.esy.es) and tern on the lamp system will send a signal 1 to processing software to change it's signal from L(low) to H(high) , After that processing software will allow Arduino to get the value of temperature from LM35. If the temperature exceed from the range that allowed Arduino will send signal to the motor to run. When we turn off the lamp the opposite will happened. If the temperature was in the range the motor will not run even if we turn the lamp on.

3.4 – Codes:

3.4.1 – Code of processing sketch:

```
import processing.serial.*;
 Serial port;


 void setup()  {
 /*  change the number between the brackets after Serial.list() to meet the
PORT number that arduino is connected to */


    port = new Serial(this, "com38", 9600);
}
 void draw() {


  String LED_Status[] =
loadStrings("http://temperaturecontrol.esy.es//LED.txt");
  print(LED_Status[0]);


  if (LED_Status[0].equals("1") == true) {
   println("LED.txt have ( 1 )- sending 'H' to Arduino ");

   port.write('H');

 }


 else {

   println("LED.txt have ( 0 )- sending 'L' to Arduino ");

   port.write('L');

 }


  delay(5000); // This makes a 5 sec delay between each check for the
value that is stored in LED.txt

 }
```

3.4.2 – Code of arduino sketch:

```
float temp;

int tempPin = 0;

int Led =13;

int Led1 = 12;  // led is connected to pin number 12

int incomingByte; // variable to read the incoming serial data coming
from "Processing" sketch through USB cable
```

```
void setup() {
  Serial.begin(9600);
  pinMode(Led, OUTPUT);
}
void loop() {
  temp = analogRead(tempPin);
  temp = temp * 0.48828125;
 Serial.print("TEMPRATURE = ");
  Serial.print(temp);
  Serial.print("*C");
  Serial.println();
  delay(1000);
  if (Serial.available() > 0) {  // checking if there is incoming serial data


    incomingByte = Serial.read();   // reading the incoming data from
"Processing"


    if (incomingByte == 'H') { // if "Processing"" is sending H , turn on the
LED
      digitalWrite(Led1, HIGH);   }
  if(temp<32){
digitalWrite(Led, LOW);
}
else{
digitalWrite(Led, HIGH);
}
    if (incomingByte == 'L') {  // if "Processing" is sending L , turn off the
LED
      digitalWrite(Led1, LOW);
  digitalWrite(Led, LOW);  }
  }
}
```

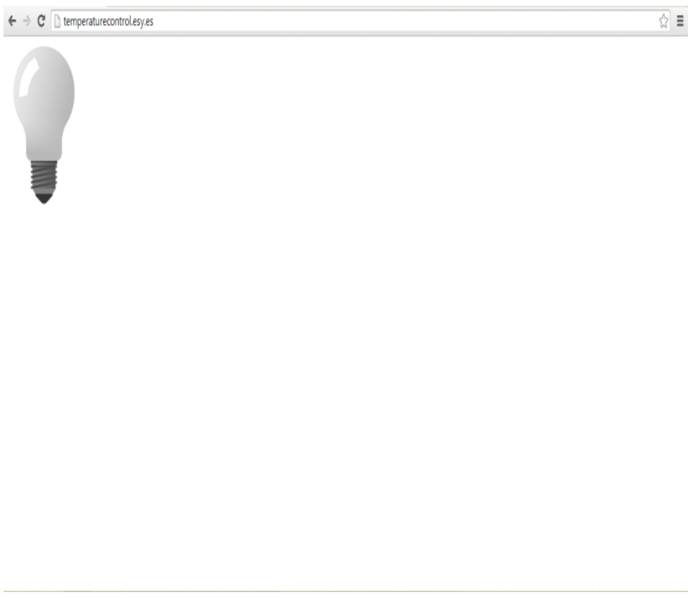
CHAPTER FOUR

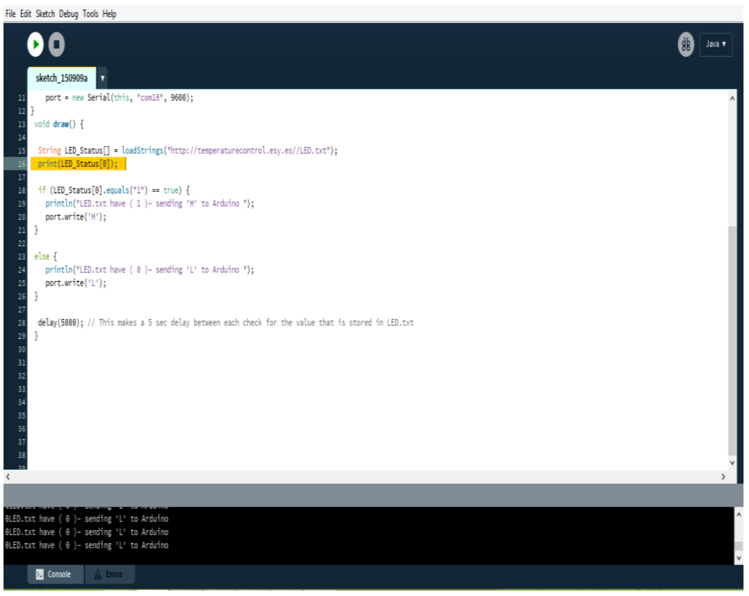
DISCUSSION AND RESULTS

CHAPTER FOUR

4.1 – Results:

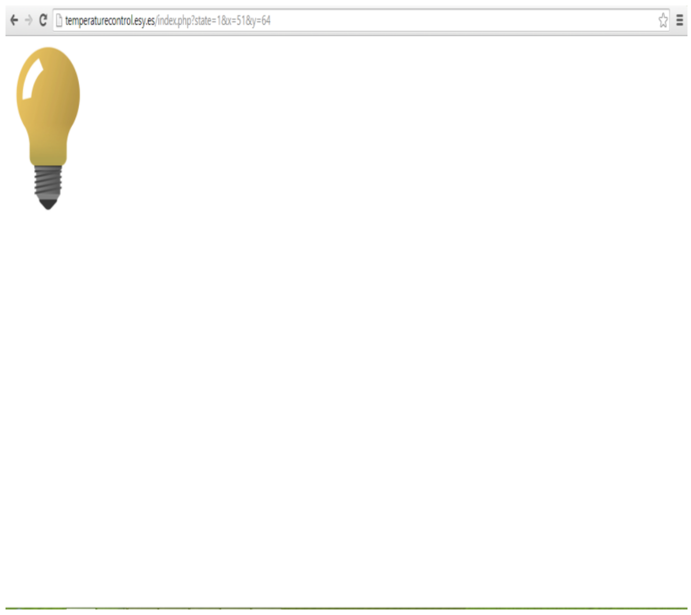
Processing software is connected directly with the Internet when the page of the project opens and Lit The lamp processing change it's signal from low to high and vice versa.

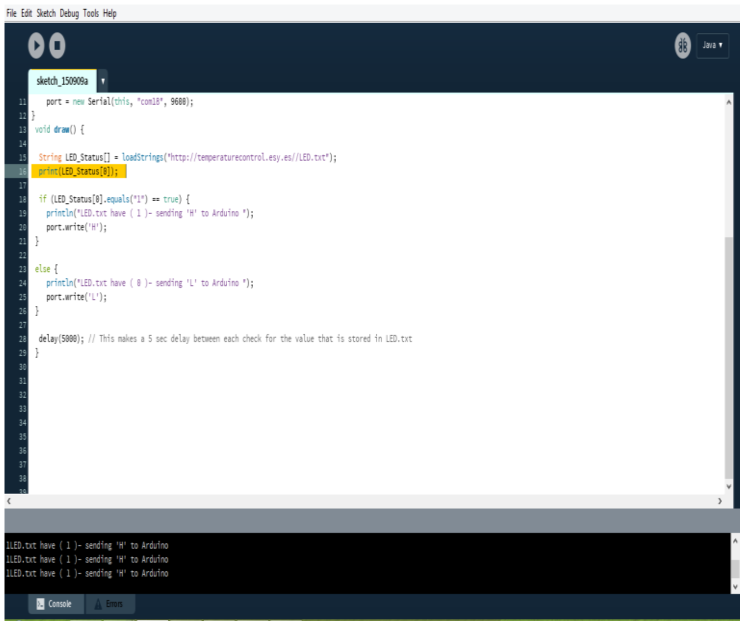




```
  port = new Serial(this, "com18", 9600);
}
void draw() {

 String LED_Status[] = loadStrings("http://temperaturecontrol.esy.es//LED.txt");
 print(LED_Status[0]);

 if (LED_Status[0].equals("1") == true) {
   println("LED.txt have ( 1 )- sending 'H' to Arduino ");
   port.write('H');
 }

 else {
   println("LED.txt have ( 0 )- sending 'L' to Arduino ");
   port.write('L');
 }

 delay(5000); // This makes a 5 sec delay between each check for the value that is stored in LED.txt
 }
```

```
0LED.txt have ( 0 )- sending 'L' to Arduino
0LED.txt have ( 0 )- sending 'L' to Arduino
0LED.txt have ( 0 )- sending 'L' to Arduino
```



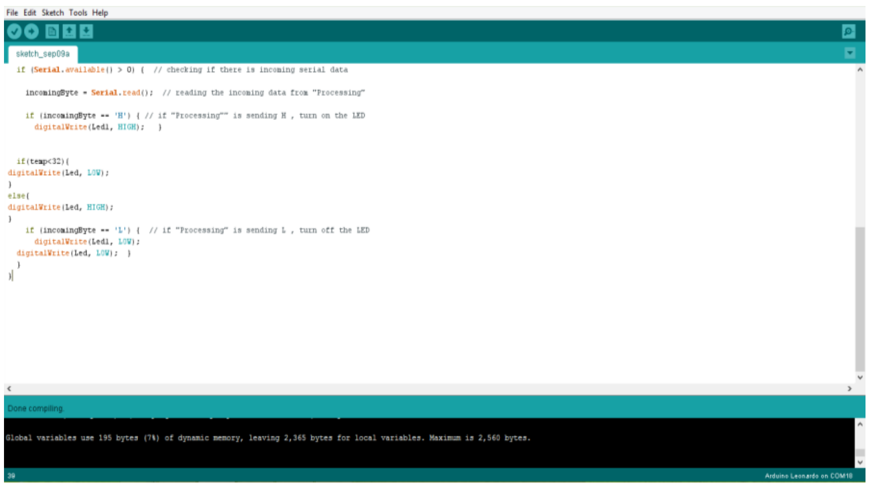


```
  port = new Serial(this, "com18", 9600);
}
void draw() {

 String LED_Status[] = loadStrings("http://temperaturecontrol.esy.es//LED.txt");
 print(LED_Status[0]);

 if (LED_Status[0].equals("1") == true) {
   println("LED.txt have ( 1 )- sending 'H' to Arduino ");
   port.write('H');
 }

 else {
   println("LED.txt have ( 0 )- sending 'L' to Arduino ");
   port.write('L');
 }

 delay(5000); // This makes a 5 sec delay between each check for the value that is stored in LED.txt
 }
```

```
1LED.txt have ( 1 )- sending 'H' to Arduino
1LED.txt have ( 1 )- sending 'H' to Arduino
1LED.txt have ( 1 )- sending 'H' to Arduino
```

Fig 4-1 Results in Processing Sketch

Processing works as governor of the project which allows the Arduino to read the temperature of the LM35 sensor And to act upon it, If temperature was within permissible limits, the motor will not work, and if it was over the permissible limits, the motor will work automatically until The lamp closed from home on the internet or the temperature drops to permissible limits.



```
  if (Serial.available() > 0) {  // checking if there is incoming serial data

    incomingByte = Serial.read();  // reading the incoming data from "Processing"

    if (incomingByte == 'H') { // if "Processing"" is sending H , turn on the LED
      digitalWrite(Led1, HIGH);   }


  if(temp<32){
digitalWrite(Led, LOW);
}
else{
digitalWrite(Led, HIGH);
}
    if (incomingByte == 'L') {  // if "Processing" is sending L , turn off the LED
      digitalWrite(Led1, LOW);
  digitalWrite(Led, LOW);  }
   }
}
```

```
Done compiling.
Global variables use 195 bytes (7%) of dynamic memory, leaving 2,365 bytes for local variables. Maximum is 2,560 bytes.
39                                                                  Arduino Leonardo on COM18
```

Fig 4-2 Arduino Sketch

It has been applied the above in practice and it has been done very satisfactorily , as we tried to send the signal from another computer and the system works with high efficiency.

The motor is in charge for the operation of the compressor which is the heart of the air conditioning and refrigeration systems.

4.2 – Response times:

The response time or delay defines how long it takes for the command to completely arrive at the destination[5]. The following table shows the differences in delay times which depend on the method of transferring the command (WLAN and Internet).

Table 4-1:The response times

WLAN	Internet
2-13 ms	5-128 ms

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

CHAPTER FIVE

5.1 – Conclusion:

An approach to control over client/server technique has been delivered in this project provides that the temperature was the object under control, the system composed of GUI's that provide an obvious way to issue the commands to the desired object, the knowledge of the underlying infrastructure is not required to achieve the process of transporting the commands.

Most of systems implementation impose two constraints in general the first is the cost, the latter is performance. The implementation of the system has been developed using a wired interface to the server computer as opposed to the researched wireless recommended approach due to high cost for wireless modules, the performance in this project was assessed by time response, the time response from initial results is affected greatly by the used connection type and it's speed , but it's still possible to control the temperature to a high reliability

5.2 – Recommendations:

There are some features should be considered for the future work such as:

1. Enhance the power supply by adding high duty mountable rechargeable batteries.
2. Using Arduino shields or Raspberry pi to develop the circuit.
3. Develop a web-based method.

4. Other parameters can be controlled (humidity, pressure, etc.)
5. It is not necessary that the motor controls the compressor only. Motor can be controlled gates channels.
6. This project can be developed for use in power plants.

REFERENCES:

[1]. COLAK,I., DEMIRBAS,S., SEFA,I., IRMAK, E. & KAHRAMAN, H. T. 2008. Remote controlling and monitoring of HVAC system over internet. Journal of Scientific and Industrial Research,67, 680.

[2]. ASHARI, A. 2010. Distributed Monitoring and Controlling Using Microcontroller and Virtual Internet Protocol. Telkomnika,8.

[3]. AHNN, J. H. 2007. The Robot control using the wireless communication and the serial communication. Cornell University.

[4]. NİHAN GÖK 7328, Sep 2006**,** GAS BOILER CARD **(**KOMBI**)**.Available**:** https://fensware.sabanciuniv.edu/ugprojects/ens4912.php?term=10. (accessed 17/4/2015).

[5]. FOROUZAN, A. B. 2006. Data Communications & Networking (sie) ,Tata McGraw-Hill Education.

[6]. ENCYCLOPEDIA, W. DC Motor [online].Available: http://en.wikipedia.org/wiki/DC motor .

(accessed).

[7]. Arduino Software Available: https://www.arduino.cc/en/Guide/Environment

(accessed 23/6/2015)

[8]. The Definition Of Diode:

http://whatis.techtarget.com/definition/diode

(accessed 24/6/2015)

[9]. Understanding Electronic Components (Transistors):

http://www.mikroe.com/old/books/keu/04.htm

http://www.infoplease.com/encyclopedia/science/transistor-types-transistors.html

(accessed 24/6/2015)

[10]. Temperature sensor lm35 Datasheet:

http://www.engineersgarage.com/electronic-components/lm35-sensor-datasheet

(accessed 24/6/2015)

[11]. Processing Software Available: https://processing.org/reference/environment/ .

(accessed 24/6/2015)

.