

PARALYSIS PATIENT HEALTH CARE KIT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology in Biotechnology

By

Bhuvanesh.P(39230007)

Mohan Raj.M(39230027)

Rohith.R.K(39230036)

Sanjay.M(39230039)



DEPARTMENT OF BIOTECHNOLOGY
SCHOOL OF BIO AND CHEMICAL

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF BIOTECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of

Bhuvanesh.P(39230007),MohanRaj.M(39230027),Rohith.R.K(39230036),

Sanjay.M(39230039) who carried out the project entitled **PARALYSIS PATIENT**

HEALTH CARE KIT under our supervision from to

Internal Guide
DR.JAYASHREE
NELLORE

External Guide
MR.PREMKUMAR

Head of the Department
DR.RAMESH KUMAR

Submitted for Viva voce Examination held on 29.04.2022

Internal Examiner

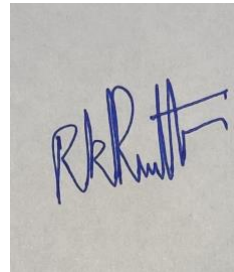
External Examiner

DECLARATION

I **R.K.Rohith(39230036)** hereby declare that the Project Report entitled **Paralysis Patient Health Care Kit** is done by me under the guidance of Dr.Jayashree Nellore(Internal) and Mr.Premkumar(External) at **Sathyabama Institute of Science and Technology** JEPPIARNAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Biotechnology.

DATE: 28.04.2022

PLACE: Chennai

A square image showing a handwritten signature in blue ink. The signature appears to be 'RKR' followed by a stylized flourish.

Signature of the Candidate

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr.Ramesh Kumar., Dean, School of Bio and Chemical and Dr.Ramesh Kumar, Head of the Department, Dept. Of Biotechnology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr.Jayashree Nellore for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Biotechnology who were helpful in many ways for the completion of the project.

ABSTRACT

The IOT based paralysis patient health care system is a system designed to help the patient convey various messages to doctors, nurse, or his/her loved ones sitting at home or office over the internet. The system makes use of a microcontroller based circuitry to achieve this functionality. It makes use of a hand motion recognition circuit and a receiver plus transmitter circuit. The hand motion circuit is used to detect hand movements using accelerometer+gyro and then transmit this information wirelessly over rf to the receiver system. The receiver system is designed to receive and process these commands and display them over the LCD display as well as transmit the data online over to IOT gecko server. The IOTgecko server then displays this information online, to achieve the desired output.

CONTENTS

S.No	Topics	Page No
1	Introduction	1
2	Project Block Diagram	4
3	Objectives	4
4	Hardware Requirements	5
5	Software Specifications	12
6	Coding	25
7	Hardware Testing	30
8	Outcomes	31
9	Result	32
10	Conclusion	36
11	References	36

1. INTRODUCTION

1.1 PARALYSIS

Paralysis is the loss of muscle function in part of your body. It happens when something goes wrong with the way messages pass between your brain and muscles. Paralysis can be complete or partial. It can occur on one or both sides of your body. It can also occur in just one area, or it can be widespread. Paralysis of the lower half of your body, including both legs, is called paraplegia. Paralysis of the arms and legs is quadriplegia.

Most paralysis is due to strokes or injuries such as spinal cord injury or a broken neck. Other causes of paralysis include:

- Nerve diseases such as amyotrophic lateral sclerosis.
- Autoimmune diseases such as Guillain-Barre syndrome.
- Bell's palsy, which affects muscles in the face.

Someone who is paralyzed because of a birth defect or sudden injury often can't feel or move anything at all in their affected body parts. Someone paralyzed by a medical condition, like multiple sclerosis (MS), might feel tingling or muscle weakness.

Paralysis can cause problems with blood flow, breathing, how well your organs work, speaking or swallowing, sexual responses, or controlling the urge to go to the bathroom, depending on where you're paralyzed and how bad it is.

1.1.2 TYPES OF PARALYSIS

- ✧ **Complete paralysis** is when you can't move or control your paralyzed muscles at all. You also may not be able to feel anything in those muscles.
- ✧ **Partial or incomplete paralysis** is when you still have some feeling in, and possibly control over, your paralyzed muscles. This is sometimes called **paresis**.
- ✧ **Localized paralysis** affects just one specific area, like your face, hands, feet, or vocal cords.
- ✧ **Generalized paralysis** is more widespread in your body and is grouped by how much of your body is affected. The type usually depends on where your brain or spinal cord is injured.

1.1.2.1 TYPES OF GENERALIZED PARALYSIS

- ✓ **Monoplegia** is a kind of generalized paralysis that affects just one limb.
Diplegia affects the same area on both sides, like both arms, both legs, or both sides of your face.
- ✓ **Hemiplegia** affects just one side of your body and is usually caused by a stroke, which damages one side of your brain.
- ✓ **Quadriplegia** (or tetraplegia) is when all four limbs are paralyzed, sometimes along with certain organs.
- ✓ **Paraplegia** is paralysis from the waist down.
- ✓ **Locked-in syndrome** is the rarest and most severe form of paralysis, where a person loses control of all their muscles except the ones that control their eye movements.

1.1.3 CAUSES OF PARALYSIS

Paralysis is most often caused by damage in the nervous system, especially the spinal cord. Other major causes are stroke, trauma with nerve injury, poliomyelitis, cerebral palsy, peripheral neuropathy, Parkinson's disease, ALS, botulism, spina bifida, multiple sclerosis, and Guillain–Barré syndrome. Temporary paralysis occurs during REM sleep, and dysregulation of this system can lead to episodes of waking paralysis. Drugs that interfere with nerve function, such as curare, can also cause paralysis.

1.2 INTERNET OF THINGS

The term *Internet of Things* (often abbreviated *IoT*) was coined more than ten years ago by industry researchers but has emerged into mainstream public view only more recently. Some claim the Internet of Things will completely transform how computer networks are used for the next 10 or 100 years, while others believe IoT is simply hype that won't much impact the daily lives of most people.

Internet of Things represents a general concept for the ability of network devices to sense and collect data from the world around us, and then share that data across the Internet where it can be processed and utilized for various interesting purposes.

Some also use the term *industrial Internet* interchangeably with IoT. This refers

primarily to commercial applications of IoT technology in the world of manufacturing. The Internet of Things is not limited to industrial applications, however.

1.2.1 What the Internet of Things Can Do for Us

Some future consumer applications envisioned for IoT sound like science fiction, but some of the more practical and realistic sounding possibilities for the technology include:

- ◆ Receiving warnings on your phone or wearable device when IoT networks detect some physical danger is detected nearby
- ◆ Self-parking automobiles
- ◆ Automatic ordering of groceries and other home supplies
- ◆ Automatic tracking of exercise habits and other day-to-day personal activity including goal tracking and regular progress reports

1.2.2 Network Devices and the Internet of Things

All kinds of ordinary household gadgets can be modified to work in an IoT system. Wi-Fi network adapters, motion sensors, cameras, microphones and other instrumentation can be embedded in these devices to enable them for work in the Internet of Things. Home automation systems already implement primitive versions of this concept for things like light bulbs, plus other devices like wireless scales and wireless blood pressure monitors that each represent early examples of IoT gadgets. Wearable computing devices like watches and glasses are also envisioned to be key components in future IoT systems.

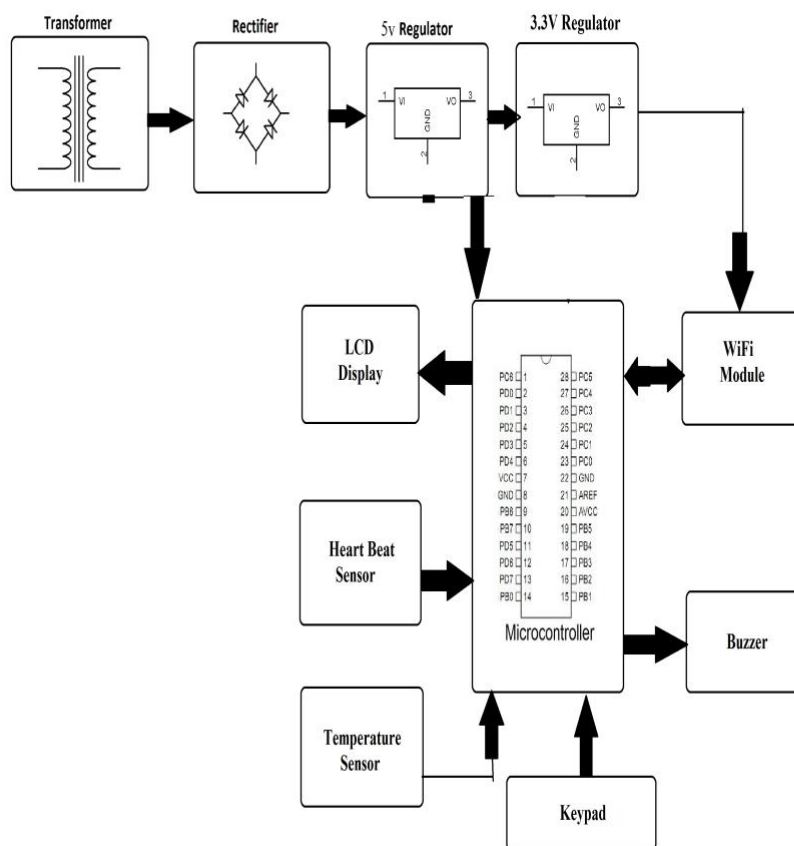
The same wireless communication protocols like Wi-Fi and Bluetooth naturally extend to the Internet of Things also.

1.2.3 ISSUES AROUND INTERNET OF THINGS

Internet of Things immediately triggers questions around the privacy of personal data. Whether real-time information about our physical location, or updates about our weight and blood pressure that may be accessible by our health care providers, having new kinds and more detailed data about ourselves streaming over wireless networks and potentially around the world is an obvious concern.

Supplying power to this new proliferation of IoT devices and their network connections can be expensive and logistically difficult. Portable devices require batteries that someday must be replaced. Although many mobile devices are optimized for lower power usage, energy costs to keep potentially billions of them running remains high.

2. PROJECT BLOCK DIAGRAM



3. OBJECTIVES

- To develop new innovation for the use of basic nursing care.
- To reduce workload on nursing staff.
- To reduce unnecessary expenditure.
- To communicate with the care taker at any time.

4.HARDWARE REQUIREMENTS

- Atmega Microcontroller
- Accelerometer & Gyro
- ESP8266 Wifi Module
- RF Tx Rx
- LCD Display
- Crystal Oscillator
- Resistors
- Capacitors
- Transistors
- Cables and Connectors
- Diodes
- PCB and Breadboards
- LED
- Transformer/Adapter
- Push Buttons
- Switch
- IC
- IC Sockets

4.1 ATMEGA MICROCONTROLLER

(PCINT14/RESET) PC6	1	[A5]	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2[0] ^{RX}	[A4]	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3[1] ^{TX}	[A3]	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4[2]	[A2]	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5[3]~	[A1]	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6[4]	[A0]	23	PC0 (ADC0/PCINT8)
VCC	7		22	GND
GND	8		21	AREF
(PCINT6/XTAL1/TOSC1) PB6	9		20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	[13]	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11[5]~	[12]	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12[6]~	[11]	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13[7]	[10]	16	PB2 ($\overline{SS}$ /OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14[8]	[9]	15	PB1 (OC1A/PCINT1)

~ = PWM

The Atmel ATmega328P is a 32K 8-bit microcontroller based on the AVR architecture. Many instructions are executed in a single clock cycle providing a throughput of almost 20 MIPS at 20MHz. The ATMEGA328-PU comes in an PDIP 28 pin package and is suitable for use on our 28 pin AVR Development Board.

The computer on one hand is designed to perform all the general purpose tasks on a single machine like you can use a computer to run a software to perform calculations or you can use a computer to store some multimedia file or to access internet through the browser, whereas the microcontrollers are meant to perform only the specific tasks, for e.g., switching the AC off automatically when room temperature drops to a certain defined limit and again turning it ON when temperature rises above the defined limit.

There are number of popular families of microcontrollers which are used in different applications as per their capability and feasibility to perform the desired task, most common of these are 8051, AVR and PIC microcontrollers. In this we will introduce you with AVR family of microcontrollers.

4.1.1 Features include:

- High Performance, Low Power Design
- 8-Bit Microcontroller Atmel® AVR® advanced RISC architecture
- 131 Instructions most of which are executed in a single clock cycle
 - I. Up to 20 MIPS throughput at 20 MHz
 - II. 32 x 8 working registers
 - III. 2 cycle multiplier
- Memory Includes
 - I. 32KB of of programmable FLASH
 - II. 1KB of EEPROM
 - III. 2KB SRAM
 - IV. 10,000 Write and Erase Cycles for Flash and 100,000 for EEPROM
 - V. Data retention for 20 years at 85°C and 100 years at 25°C
 - VI. Optional boot loader with lock bits
- In System Programming (ISP) by via boot loader
- True Read-While-Write operation
- Features Include
 - I. 2 x 8-bit Timers/Counters each with independent prescaler and compare modes
 - II. A single 16-bit Timer/Counter with an independent prescaler, compare and capture modes
 - III. Real time counter with independent oscillator
 - IV. 10 bit, 6 channel analog to digital Converter

- V. 6 pulse width modulation channels
- VI. Internal temperature sensor
- VII. Serial USART (Programmable)
- VIII. Master/Slave SPI Serial Interface - (Philips I2C compatible)
- IX. Programmable watchdog timer with independent internal oscillator
- X. Internal analog comparator
- XI. Interrupt and wake up on pin change

■ Additional Features

- I. Internal calibrated oscillator
- II. Power on reset and programmable brown out detection
- III. External and internal interrupts
- IV. 6 sleep modes including idle, ADC noise reduction, power save, power down, standby, and extended standby

■ I/O and Package

- I. 23 programmable I/O lines
- II. 28 pin PDIP package

In our days, there have been many advancement in the field of Electronics and many cutting edge technologies are being developed every day, but still 8 bit microcontrollers have its own role in the digital electronics market dominated by 16-32 & 64 bit digital devices. Although powerful microcontrollers with higher processing capabilities exist in the market, 8bit microcontrollers still hold its value because of their easy-to-understand-operation, very much high popularity, ability to simplify a digital circuit, low cost compared to features offered, addition of many new features in a single IC and interest of manufacturers and consumers.

Today's microcontrollers are much different from what it were in the initial stage, and the number of manufacturers are much more in count than it was a decade or two ago. At present some of the major manufacturers are Microchip (publication: PIC microcontrollers), Atmel (publication: AVR microcontrollers), Hitachi, Phillips, Maxim, NXP, Intel etc. Our interest is upon ATmega32. It belongs to Atmel's AVR series micro controller family. Let's see the features.

PIN count: Atmega32 has got 40 pins. Two for Power (pin no.10: +5v, pin no. 11: ground), two for oscillator (pin 12, 13), one for reset (pin 9), three for providing necessary power and reference voltage to its internal ADC, and 32 (4×8) I/O pins.

About I/O pins: ATmega32 is capable of handling analogue inputs. Port A can be used as either DIGITAL I/O Lines or each individual pin can be used as a single input channel to the internal ADC of ATmega32, plus a pair of pins AREF, AVCC & GND (refer to ATmega32 datasheet) together can make an ADC channel.

No pins can perform and serve for two purposes (for an example: Port A pins cannot work as a Digital I/O pin while the Internal ADC is activated) at the same time. It's the

programmers responsibility to resolve the conflict in the circuitry and the program. Programmers are advised to have a look to the priority tables and the internal configuration from the datasheet.

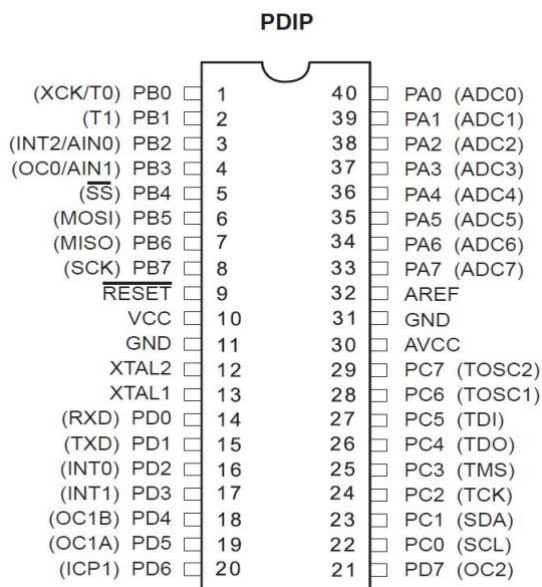
Digital I/O pins: ATmega32 has 32 pins (4portsx8pins) configurable as Digital I/O pins.

Timers: 3 Inbuilt timer/counters, two 8 bit (timer0, timer2) and one 16 bit (timer1).

ADC: It has one successive approximation type ADC in which total 8 single channels are selectable. They can also be used as 7 (for TQFP packages) or 2 (for DIP packages) differential channels. Reference is selectable, either an external reference can be used or the internal 2.56V reference can be brought into action. There external reference can be connected to the AREF pin.

Communication Options: ATmega32 has three data transfer modules embedded in it. They are

- Two Wire Interface
- USART
- Serial Peripheral Interface



Analog comparator: On-chip analog comparator is available. An interrupt is assigned for different comparison result obtained from the inputs.

External Interrupt: 3External interrupt is accepted. Interrupt sense is configurable.

Memory: It has 32Kbytes of In-System Self-programmable Flash program

memory, 1024 Bytes EEPROM, 2Kbytes Internal SRAM. Write/Erase Cycles: 10,000 Flash / 100,000 EEPROM.

Clock: It can run at a frequency from 1 to 16 MHz. Frequency can be obtained from external Quartz Crystal, Ceramic crystal or an R-C network. Internal calibrated RC oscillator can also be used.

More Features: Up to 16 MIPS throughput at 16MHz. Most of the instruction executes in a single cycle. Two cycle on-chip multiplication. 32×8 General Purpose Working Registers

Debug: JTAG boundary scan facilitates on chip debug.

Programming: Atmega32 can be programmed either by In-System Programming via Serial peripheral interface or by Parallel programming. Programming via JTAG interface is also possible. Programmer must ensure that SPI programming and JTAG are not be disabled using fuse bits; if the programming is supposed to be done using SPI or JTAG.

4.2 DIODES

Diodes are used to convert AC into DC these are used as half wave rectifier or full wave rectifier. Three points must be kept in mind while using any type of diode.

1. Maximum forward current capacity
2. Maximum reverse voltage capacity
3. Maximum forward voltage capacity

The number and voltage capacity of some of the important diodes available in the market are as follows:

- Diodes of number IN4001, IN4002, IN4003, IN4004, IN4005, IN4006 and IN4007 have maximum reverse bias voltage capacity of 50V and maximum forward current capacity of 1 Amp.
- Diode of same capacities can be used in place of one another. Besides this diode of more capacity can be used in place of diode of low capacity but diode of low capacity cannot be used in place of diode of high capacity. For example, in place of IN4002; IN4001 or IN4007 can be used but IN4001 or IN4002 cannot be used in place of IN4007. The diode BY125 made by company BEL is equivalent of diode from IN4001 to IN4003. BY 126 is equivalent to diodes IN4004 to 4006 and BY 127 is equivalent to diode

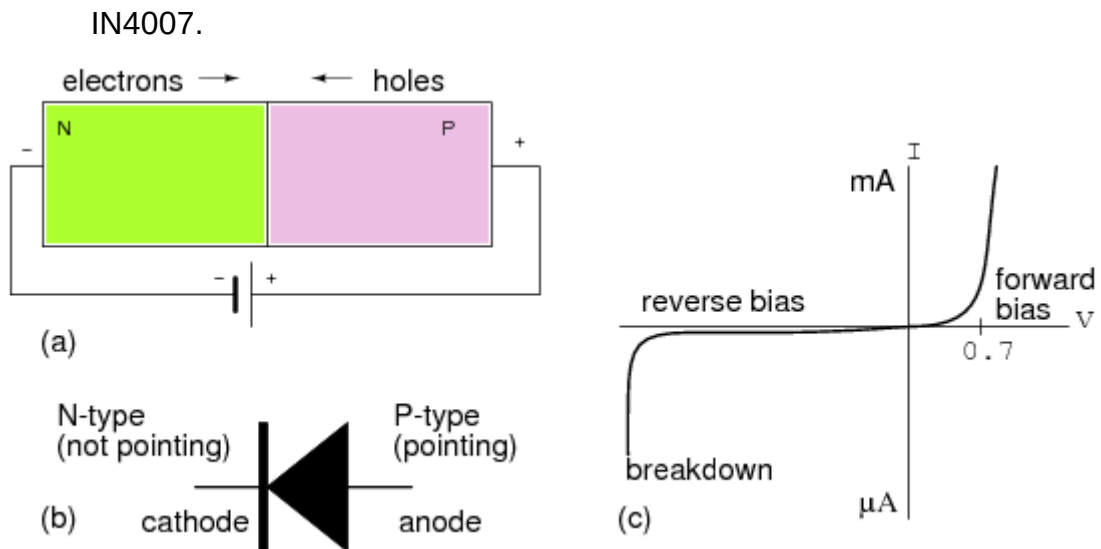


Fig:PN Junction diode

4.3 PUSH BOTTONS

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, though even many un-biased buttons (due to their physical nature) require a spring to return to their un-pushed state. Different people use different terms for the "pushing" of the button, such as press, depress, mash, and punch.

Uses:

In industrial and commercial applications push buttons can be linked together by a mechanical linkage so that the act of pushing one button causes the other button to be released. In this way, a stop button can "force" a start button to be released. This method of linkage is used in simple manual operations in which the machine or process have no electrical circuits for control.

Pushbuttons are often color-coded to associate them with their function so that the operator will not push the wrong button in error. Commonly used colors are red for stopping the machine or process and green for starting the machine or process.

Red pushbuttons can also have large heads (mushroom shaped) for easy

operation and to facilitate the stopping of a machine. These pushbuttons are called emergency stop buttons and are mandated by the electrical code in many jurisdictions for increased safety. This large mushroom shape can also be found in buttons for use with operators who need to wear gloves for their work and could not actuate a regular flush-mounted push button. As an aid for operators and users in industrial or commercial applications, a pilot light is commonly added to draw the attention of the user and to provide feedback if the button is pushed. Typically this light is included into the center of the pushbutton and a lens replaces the pushbutton hard center disk.

The source of the energy to illuminate the light is not directly tied to the contacts on the back of the pushbutton but to the action the pushbutton controls. In this way a start button when pushed will cause the process or machine operation to be started and a secondary contact designed into the operation or process will close to turn on the pilot light and signify the action of pushing the button caused the resultant process or action to start.

In popular culture, the phrase "the button" refers to a (usually fictional) button that a military or government leader could press to launch nuclear weapons.

Push to ON button:

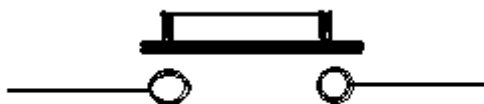


Fig 4.3(a): push on button

Initially the two contacts of the button are open. When the button is pressed they become connected. This makes the switching operation using the push button.

5.SOFTWARE SPECIFICATIONS

- ❖ Arduino Compiler
- ❖ MC Programming Language: C

5.1 ARDUINO

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

5.1.1 Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac,

Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- A. Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50
- B. Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- C. Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- D. Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C

programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

- E. Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

5.1.2 Arduino Uno

The most common version of Arduino is the Arduino Uno. This board is what most people are talking about when they refer to an Arduino. In the next step, there is a more complete rundown of its features.

5.1.2 Arduino NG, Diecimila, and the Duemilanove (Legacy Versions)

Legacy versions of the Arduino Uno product line consist of the NG, Diecimila, and the Duemilanove. The important thing to note about legacy boards is that they lack particular feature of the Arduino Uno. Some key differences:

- i. The Diecimila and NG use an ATMEGA168 chips (as opposed to the more powerful ATMEGA328)
- ii. Both the Diecimila and NG have a jumper next to the USB port and require manual selection of either USB or battery power.
- iii. The Arduino NG requires that you hold the reset button on the board for a few seconds prior to uploading a program.

5.1.3 Arduino Mega 2560

The Mega is the second most commonly encountered version of the Arduino family. The Arduino Mega is like the Arduino Uno's beefier older brother. It boasts 256 KB of memory (8 times more than the Uno). It also had 54 input and output pins, 16 of which are analog pins, and 14 of which can do PWM. However, all of the added functionality comes at the cost of a slightly larger circuit board. It may make your project more powerful, but it will also make your project larger.

5.1.4 Arduino Mega ADK

This specialized version of the Arduino is basically an Arduino Mega that has been specifically designed for interfacing with Android smartphones.

5.1.5 Arduino LilyPad

The LilyPad was designed for wearable and e-textile applications. It is intended to be sewn to fabric and connected to other sewable components using conductive thread. This board requires the use of a special FTDI-USB TTL serial programming cable. For more information, the Arduino LilyPad page is a decent starting point.

Some people think of the entire Arduino board as a microcontroller, but this is inaccurate. The Arduino board actually is a specially designed circuit board for programming and prototyping with Atmel microcontrollers.

The nice thing about the Arduino board is that it is relatively cheap, plugs straight into a computer's USB port, and it is dead-simple to setup and use (compared to other development boards).

Some of the key features of the Arduino Uno include:

- i) An open source design. The advantage of it being open source is that it has a large community of people using and troubleshooting it. This makes it easy to find someone to help you debug your projects.
- ii) An easy USB interface . The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows you to interface with it as through it were a serial device. The benefit of this setup is that serial communication is an extremely easy (and time-tested) protocol, and USB makes connecting it to modern computers really convenient.
- iii) Very convenient power management and built-in voltage regulation. You can connect an external power source of up to 12v and it will regulate it to both 5v and 3.3v. It also can be powered directly off of a USB port without any external power.
- iv) An easy-to-find, and dirt cheap, microcontroller "brain." The ATmega328 chip retails for about \$2.88 on Digikey. It has countless number of nice hardware features like timers, PWM pins, external and internal interrupts, and multiple sleep modes.
- v) A 16mhz clock. This makes it not the speediest microcontroller around, but fast

enough for most applications.

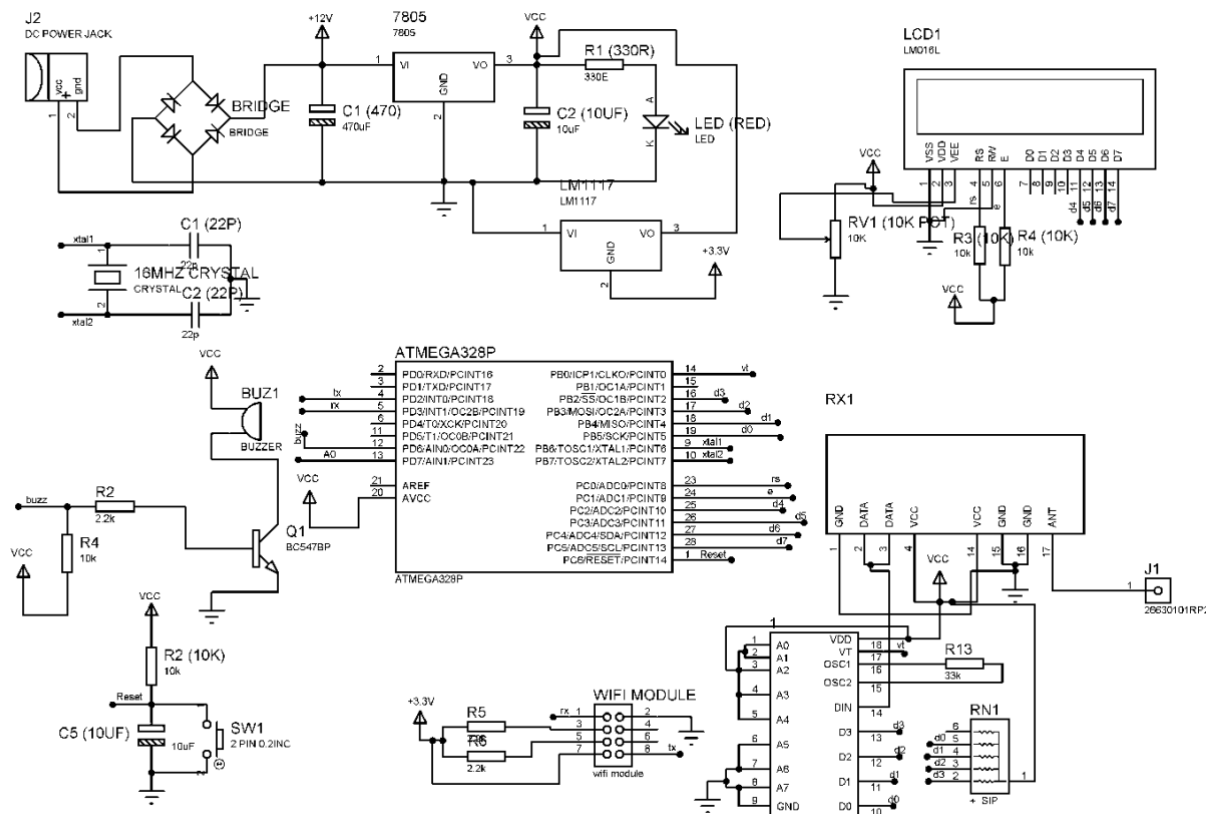
vi)32 KB of flash memory for storing your code.

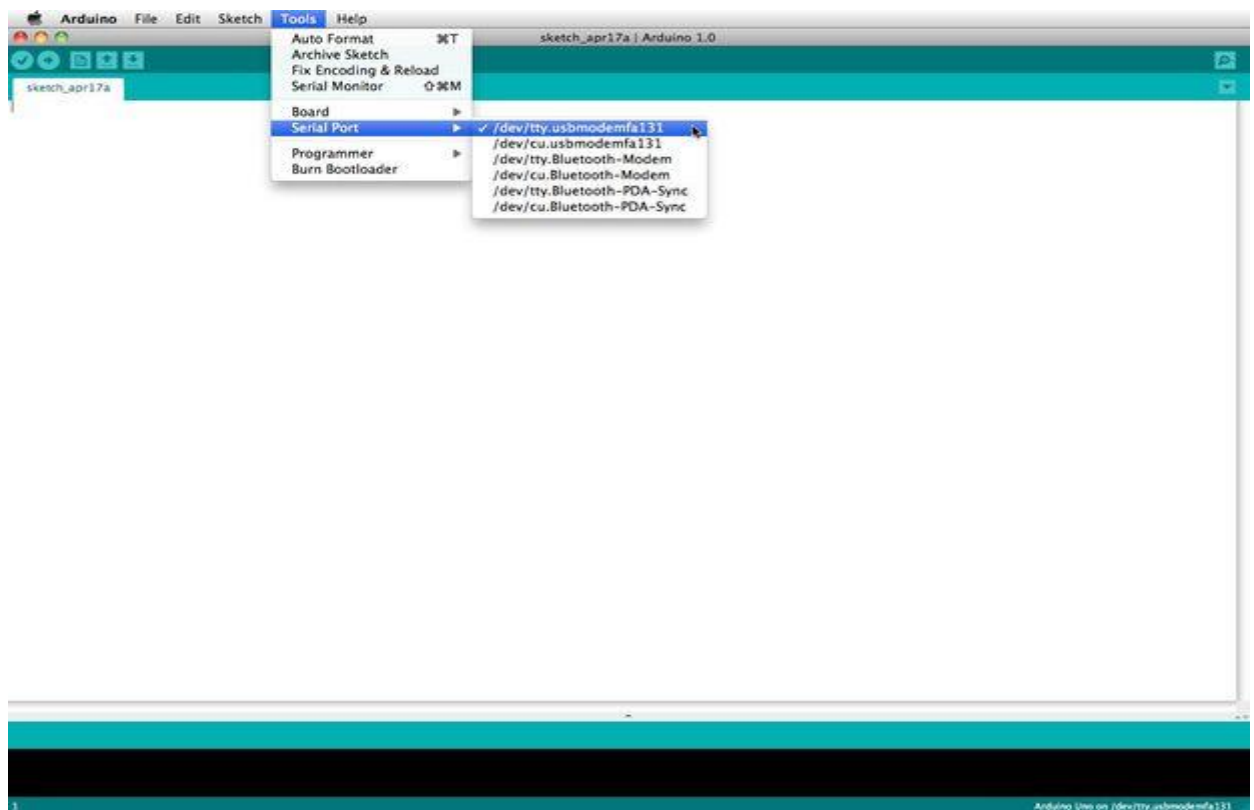
vii)13 digital pins and 6 analog pins. These pins allow you to connect external hardware to your Arduino. These pins are key for extending the computing capability of the Arduino into the real world. Simply plug your devices and sensors into the sockets that correspond to each of these pins and you are good to go.

viii)An ICSP connector for bypassing the USB port and interfacing the Arduino directly as a serial device. This port is necessary to re-bootload your chip if it corrupts and can no longer talk to your computer.

ix)An on-board LED attached to digital pin 13 for fast an easy debugging of code.

x)And last, but not least, a button to reset the program on the chip.





Before you can start doing anything in the Arduino programmer, you must set the board-type and serial port.

To set the board, go to the following:

Tools --> Boards

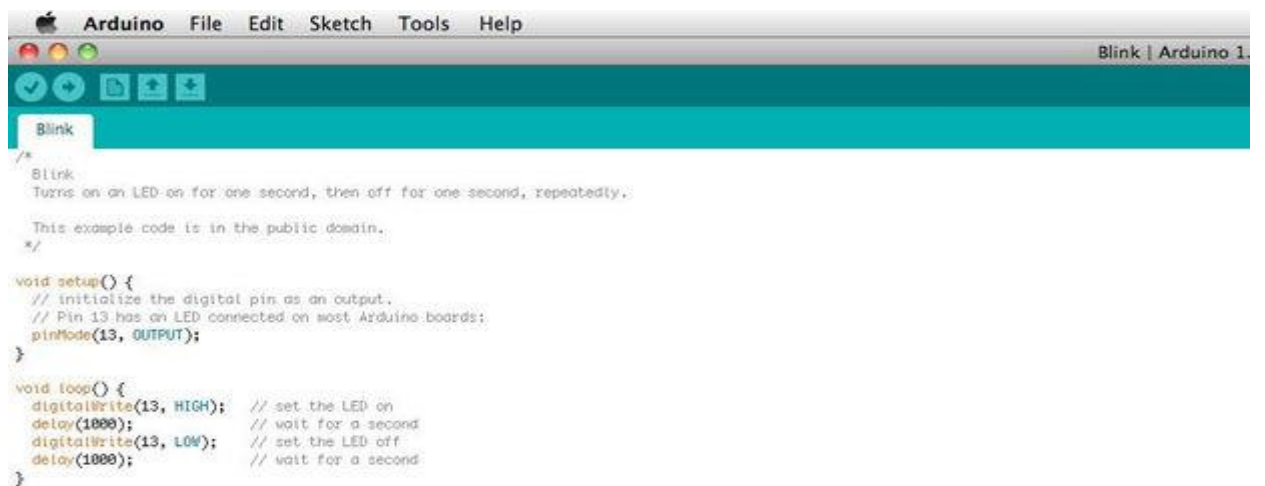
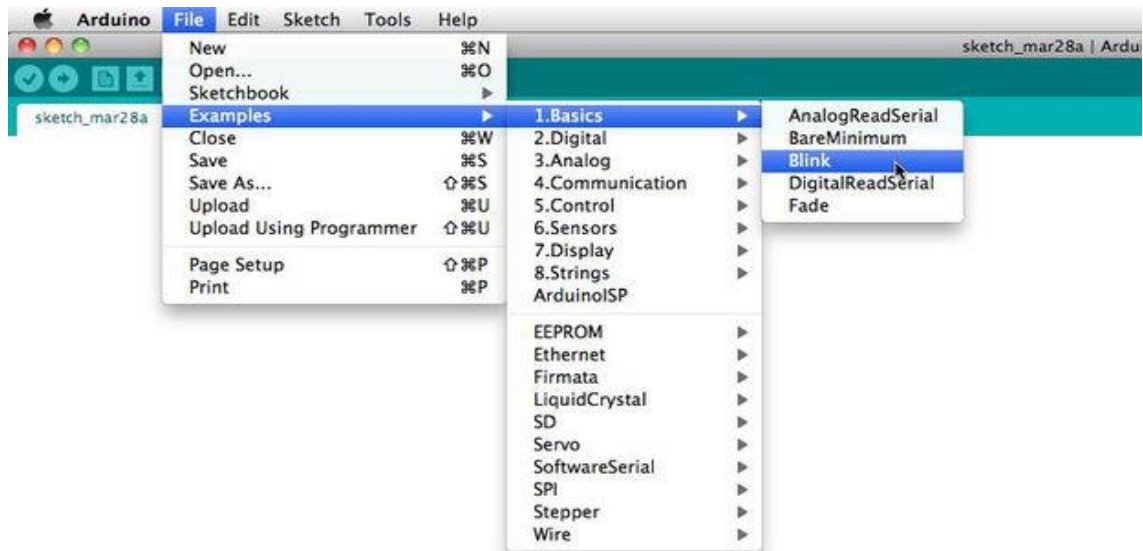
Select the version of board that you are using. Since I have an Arduino Uno plugged in, I obviously selected "Arduino Uno."

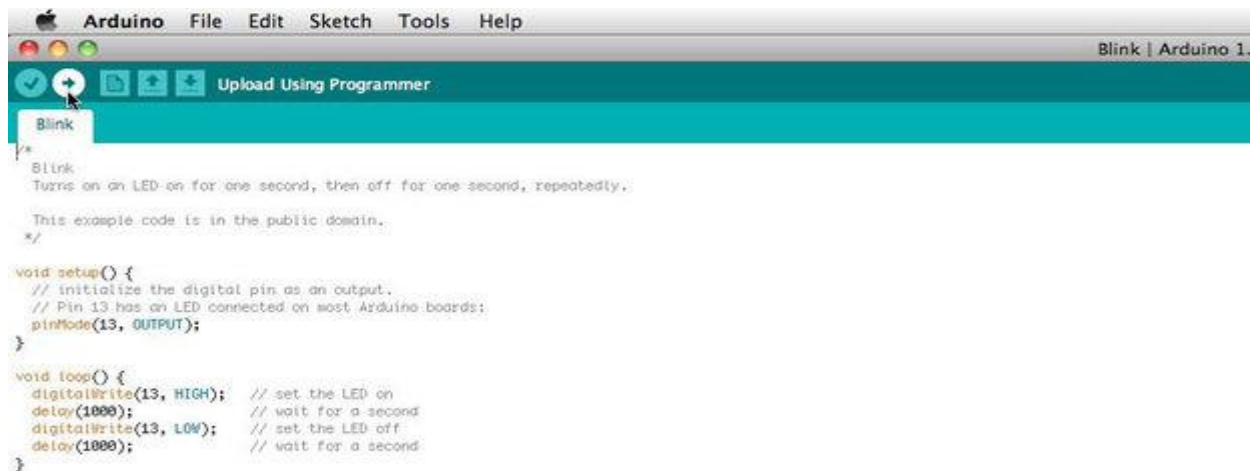
To set the serial port, go to the following:

Tools --> Serial Port

Select the serial port that looks like:

/dev/tty.usbmodem [random numbers]





Arduino programs are called sketches. The Arduino programmer comes with a ton of example sketches preloaded. This is great because even if you have never programmed anything in your life, you can load one of these sketches and get the Arduino to do something.

To get the LED tied to digital pin 13 to blink on and off, let's load the blink example. The blink example can be found here:

Files --> Examples --> Basics --> Blink

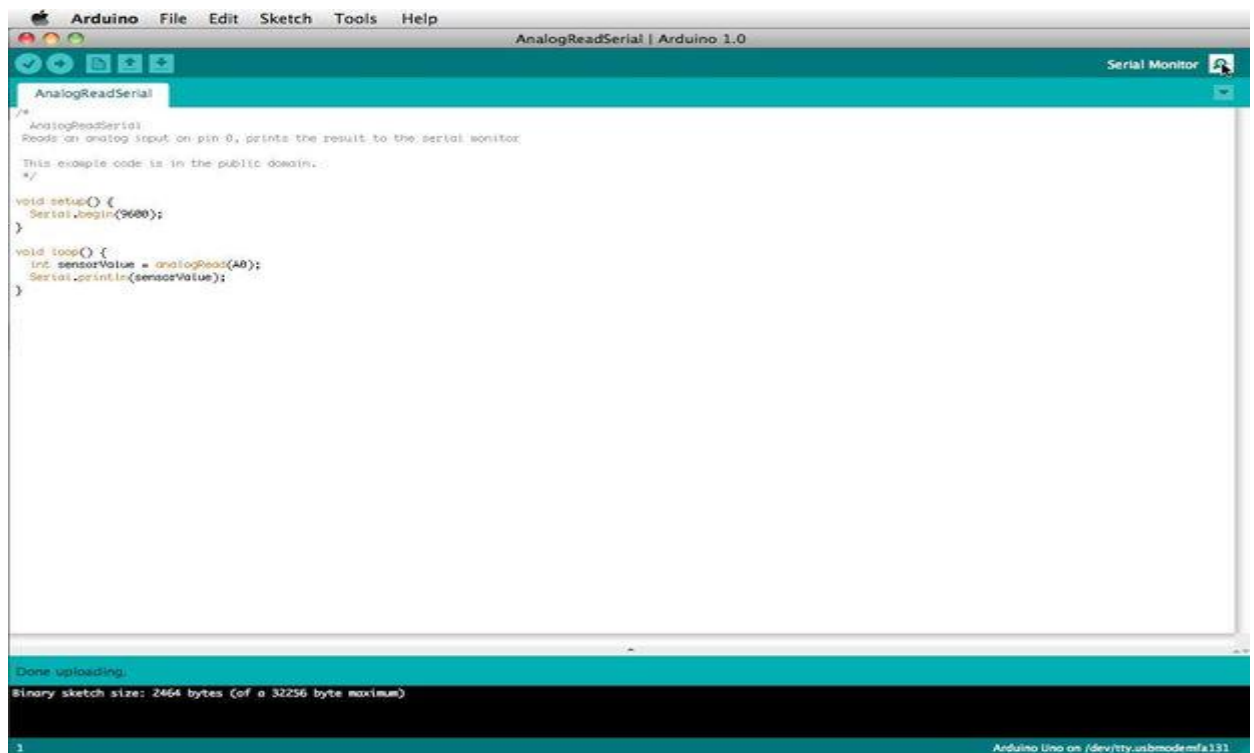
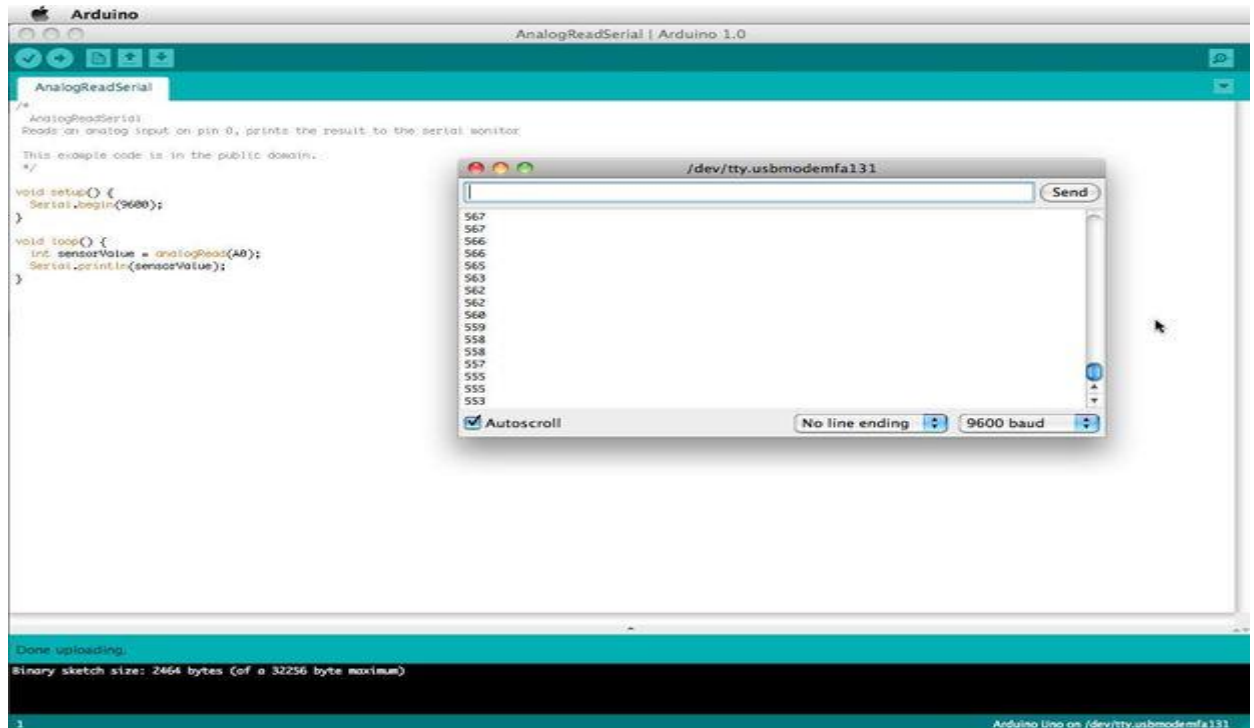
The blink example basically sets pin D13 as an output and then blinks the test LED on the Arduino board on and off every second.

Once the blink example is open, it can be installed onto the ATMEGA328 chip by pressing the upload button, which looks like an arrow pointing to the right.

Notice that the surface mount status LED connected to pin 13 on the Arduino will

start to blink. You can change the rate of the blinking by changing the length of the delay and pressing the upload button again.

Serial monitor



The serial monitor allows your computer to connect serially with the Arduino. This is important because it takes data that your Arduino is receiving from sensors and other devices and displays it in real-time on your computer. Having this ability is invaluable to debug your code and understand what number values the chip is actually receiving.

For instance, connect center sweep (middle pin) of a potentiometer to A0, and the outer pins, respectively, to 5v and ground. Next upload the sketch shown below:

File --> Examples --> 1.Basics --> AnalogReadSerial

Click the button to engage the serial monitor which looks like a magnifying glass. You can now see the numbers being read by the analog pin in the serial monitor. When you turn the knob the numbers will increase and decrease.

The numbers will be between the range of 0 and 1023. The reason for this is that the analog pin is converting a voltage between 0 and 5V to a discrete number.

The Arduino has two different types of input pins, those being analog and digital.

To begin with, let's look at the digital input pins.

Digital input pins only have two possible states, which are on or off. These two on and off states are also referred to as:

- HIGH or LOW
- 1 or 0
- 5V or 0V.

This input is commonly used to sense the presence of voltage when a switch is opened or closed.

Digital inputs can also be used as the basis for countless digital communication protocols. By creating a 5V (HIGH) pulse or 0V (LOW) pulse, you can create a binary signal, the basis of all computing. This is useful for talking to digital sensors like a PING ultrasonic sensor, or communicating with other devices.

For a simple example of a digital input in use, connect a switch from digital pin 2 to

5V, a 10K resistor** from digital pin 2 to ground, and run the following code:

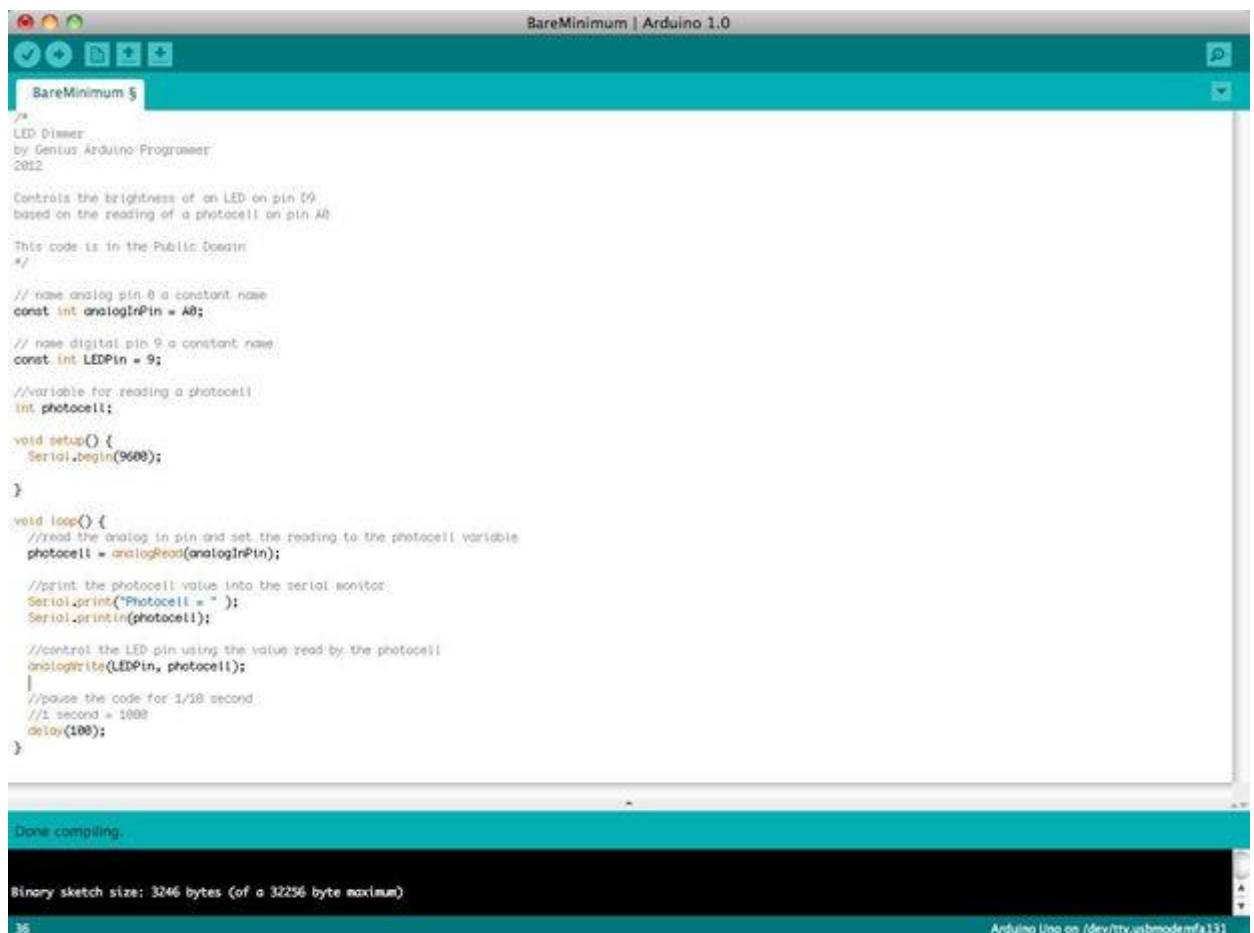
File --> Examples --> 2.Digital --> Button

5.3 ANALOG IN

Aside from the digital input pins, the Arduino also boasts a number of analog input pins.

Analog input pins take an analog signal and perform a 10-bit analog-to-digital (ADC) conversion to turn it into a number between 0 and 1023 (4.9mV steps).

This type of input is good for reading resistive sensors. These are basically sensors which provide resistance to the circuit. They are also good for reading a varying voltage signal between 0 and 5V. This is useful when interfacing with various types of analog circuitry.



```
BareMinimum 5
/*
LED Dimmer
by Genius Arduino Programmer
2012

Controls the brightness of an LED on pin 9
based on the reading of a photocell on pin A0

This code is in the Public Domain
*/

// name analog pin 0 a constant name
const int analogInPin = A0;

// name digital pin 9 a constant name
const int LEDPin = 9;

//variable for reading a photocell
int photocell;

void setup() {
  Serial.begin(9600);
}

void loop() {
  //read the analog in pin and set the reading to the photocell variable
  photocell = analogRead(analogInPin);

  //print the photocell value into the serial monitor
  Serial.print("Photocell = ");
  Serial.println(photocell);

  //control the LED pin using the value read by the photocell
  analogWrite(LEDPin, photocell);

  //pause the code for 1/10 second
  //1 second = 1000
  delay(100);
}
```

Done compiling.

Binary sketch size: 3246 bytes (of a 32256 byte maximum)

36 Arduino Uno on /dev/tty.usbmodemfa131

To write your own code, you will need to learn some basic programming language syntax. In other words, you have to learn how to properly form the code for the programmer to understand it. You can think of this kind of like understanding

grammar and punctuation. You can write an entire book without proper grammar and punctuation, but no one will be able to understand it, even if it is in English.

Some important things to keep in mind when writing your own code:

- An Arduino program is called a sketch.
 - All code in an Arduino sketch is processed from top to bottom.
 - Arduino sketches are typically broken into five parts.
 1. The sketch usually starts with a header that explains what the sketch is doing, and who wrote it.
 2. Next, it usually defines global variables. Often, this is where constant names are given to the different Arduino pins.
 3. After the initial variables are set, the Arduino begins the setup routine. In the setup function, we set initial conditions of variables when necessary, and run any preliminary code that we only want to run once. This is where serial communication is initiated, which is required for running the serial monitor.
 4. From the setup function, we go to the loop routine. This is the main routine of the sketch. This is not only where your main code goes, but it will be executed over and over, so long as the sketch continues to run.
 5. Below the loop routine, there is often other functions listed. These functions are user-defined and only activated when called in the setup and loop routine. When these functions are called, the Arduino processes all of the code in the function from top to bottom and then goes back to the next line in the sketch where it left off when the function was called. Functions are good because they allow you to run standard routines - over and over - without having to write the same lines of code over and over. You can simply call upon a function multiple times, and this will free up memory on the chip because the function routine is only written once. It also makes code easier to read. To learn how to form your own functions, check out [this page](#).
-
- ✧ All of that said, the only two parts of the sketch which are mandatory are the Setup and Loop routines.
 - ✧ Code must be written in the Arduino Language, which is roughly based on C.
 - ✧ Almost all statements written in the Arduino language must end with a ;
 - ✧ Conditionals (such as if statements and for loops) do not need a ;

- ✧ Conditionals have their own rules and can be found under "Control Structures" on the Arduino Language page
- ✧ Variables are storage compartments for numbers. You can pass values into and out of variables. Variables must be defined (stated in the code) before they can be used and need to have a data type associated with it. To learn some of the basic data types, review the Language Page.

Okay! So let us say we want to write code that reads a photocell connected to pin A0, and use the reading we get from the photocell to control the brightness of an LED connected to pin D9.

First, we want to open the BareMinimum sketch, which can be found at:

File --> Examples --> 1.Basic --> BareMinimum

The BareMinimum Sketch should look like this:

```
<pre>void setup() {  
    // put your setup code here, to run once:  
  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```

6. CODING

```
#include <SoftwareSerial.h>  
#include <LiquidCrystal.h>  
  
SoftwareSerial esp8266(3 , 2);  
const int d0 = 13;  
const int d1 = 12;  
const int d2 = 11;  
const int d3 = 10;  
const int VT = 8;  
const int buzzer = 6;  
unsigned int x = 0;  
unsigned int data_present;  
int VT_status = 0;  
int data0 = 0;
```

```

int data1 = 0;
int data2 = 0;
int data3 = 0;

#define ATE "ATE0\r\n"
#define ok "OK"
#define pinstr "pin="
#define err "Error"
#define InvalidUserIdOrPassword 18
#define InvalidData 45
#define HitLimitExceed 104
#define VALID 24
#define error18 "18"
#define error45 "45"
#define error104 "104"
#define CIPSEND "AT+CIPSEND=4,"

#define CWJAP "AT+CWJAP=\"iot\", \"project1234\"\\r\\n"

bool mode_status;
String dataOne = "SOME%20PROBLEM";
String dataTwo = "Call%20Attendant";
String dataThree = "I%20Need%20food";
String dataFour = "I%20Need%20water";
String dataFive = "No%20Input";
bool pump_value = 1;
unsigned int flame_value;
unsigned int flame_valuel;
int err_index;
String finallinktoSenddata;
String data;
String linkTogetsiteResponse = "";
String linkTosendData = "";
String lenghtOflinstr;
const String id = "jhachandan186@gmail.com"; // iotgecko login id
const String pass = "5582"; // iotgecko login password
//String ssid = "iot"; // SSID/name of your wifi router or wifi hotspot
//String pass_key = "project1234"; // Wifi password
String CIPSTART = "AT+CIPSTART=4, \"TCP\", \"www.iotgecko.com\", 80\r\n";
String Host = " HTTP/1.1\r\nHost: www.iotgecko.com\r\n\r\n";
bool notConected = true;
bool login = false;
const int no_of_senddata = 1;
const int no_of_getdata = 0;
String iot_data[no_of_senddata];
char motor_status;
char motor_status1;
unsigned long current_time ;

int pwm_val = 0;
char buff[15];
bool iot_control = true;
bool data_available;
int data1_no = 0;
int data2_no = 0;
int data3_no = 0;
int data4_no = 0;

```

```

LiquidCrystal lcd(A0, A1, A2, A3, A4, A5); //RS, EN, D4, D5, D6, D7

void setup() {
  // put your setup code here, to run once:
  esp8266.begin(9600);
  Serial.begin(9600);

  pinMode(VT, INPUT );
  pinMode(d0, INPUT );
  pinMode(d1, INPUT );
  pinMode(d2, INPUT );
  pinMode(d3, INPUT );
  pinMode(buzzer, OUTPUT);
  digitalWrite(buzzer, LOW);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("IOT Automated");
  lcd.setCursor(0, 1);
  lcd.print("Paralysis System");

  esp8266.listen();
  init_wifi();
}

void loop() {
  // put your main code here, to run repeatedly:
  VT_status = digitalRead(VT);
  data0 = digitalRead(d0);
  data1 = digitalRead(d1);
  data2 = digitalRead(d2);
  data3 = digitalRead(d3);
  rf_process();

}

void rf_process()
{
  if ((VT_status == 1) && (data0 == 1) && (data1 == 0) && (data2 == 0) &&
(data3 == 0))
  {
    lcd.clear();
    lcd.print("SOME PROBLEM");

    tone_buzzer(4, 500);

    unsigned long int past_time = millis();
    do
    {
      unsigned long int present_time = millis();
      tone_buzzer(3,200);
      if((present_time -past_time)>5000)

      {
        lcd.clear();
        lcd.print("Sending Data to ");
        lcd.setCursor(6, 1);

```

```

        lcd.print("IOT");
        delay(1000);
        iot_data[0] = dataOne;
        Serial.print("data sent:");
        Serial.println(iot_data[0]);
        while ( hit_link() == 0);
        break;
    }
}
while(digitalRead(d0));

    delay(100);
}
else if ((VT_status == 1) && (data0 == 0) && (data1 == 1) && (data2 ==
0) && (data3 == 0))
{
    lcd.clear();
    lcd.print("Call attendant");

    tone_buzzer(4, 500);
    unsigned long int past_time = millis();
    do
    {
        unsigned long int present_time = millis();
        tone_buzzer(3,200);
        if((present_time -past_time)>5000)

        {
            lcd.clear();
            lcd.print("Sending Data to ");
            lcd.setCursor(6, 1);
            lcd.print("IOT");
            delay(1000);
            iot_data[0] = dataTwo;
            Serial.print("data sent:");
            Serial.println(iot_data[0]);
            while ( hit_link() == 0);
            break;
        }
    }
    while(digitalRead(d1));
    delay(100);
}
else if ((VT_status == 1) && (data0 == 0) && (data1 == 0) && (data2 ==
1) && (data3 == 0))
{
    lcd.clear();
    lcd.print("I Need Food");
    tone_buzzer(4, 500);

    unsigned long int past_time = millis();
    do
    {
        unsigned long int present_time = millis();
        tone_buzzer(3,200);
        if((present_time -past_time)>5000)

```

```

{
  lcd.clear();
  lcd.print("Sending Data to ");
  lcd.setCursor(6, 1);
  lcd.print("IOT");
  delay(1000);
  iot_data[0] = dataThree;
  Serial.print("data sent:");
  Serial.println(iot_data[0]);
  while ( hit_link() == 0);

  break;
}
}
while(digitalRead(d2));
  delay(100);
}
else if ((VT_status == 1) && (data0 == 0) && (data1 == 0) && (data2 ==
0) && (data3 == 1))
{
  lcd.clear();
  lcd.print("I Need Water");

  tone_buzzer(4, 500);

  unsigned long int past_time = millis();
  do
  {
    unsigned long int present_time = millis();
    tone_buzzer(3,200);
    if((present_time -past_time)>5000)

    {
      lcd.clear();
      lcd.print("Sending Data to ");
      lcd.setCursor(6, 1);
      lcd.print("IOT");
      delay(1000);
      iot_data[0] = dataFour;
      Serial.print("data sent:");
      Serial.println(iot_data[0]);
      while ( hit_link() == 0);

      break;
    }
  }
  while(digitalRead(d3));
    delay(100);
}

else if ( (data0 == 0) && (data1 == 0) && (data2 == 0) && (data3 == 0))
{
  lcd.clear();
  lcd.print("Display Unit");
  lcd.setCursor(0, 1);

```

```

lcd.print("No Input");
unsigned long int past_time = millis();
do
{
    unsigned long int present_time = millis();
    if((present_time -past_time)>15000)

    {

        iot_data[0] = dataFive;
        Serial.print("data sent:");
        Serial.println(iot_data[0]);
        while ( hit_link() == 0);

        break;
    }

    }while((!(digitalRead(d0))&&!(digitalRead(d1))&&!(digitalRead(d2))
&&!(digitalRead(d3))));

    delay(200);
}

}

void tone_buzzer(int i, int d)
{
    for (int j = 0; j < i; j++)
    {
        digitalWrite(buzzer, HIGH);
        delay(d);
        digitalWrite(buzzer, LOW);
        delay(50);
    }
}

```

7. HARDWARE TESTING

7.1 CONTINUITY TEST

In electronics, a continuity test is the checking of an electric circuit to see if current flows (that it is in fact a complete circuit). A continuity test is performed by placing a small voltage (wired in series with an LED or noise-producing component such as a piezoelectric speaker) across the chosen path. If electron flow is inhibited by broken conductors, damaged components, or excessive resistance, the circuit is "open". Devices that can be used to perform continuity tests include multi meters which measure current and specialized continuity testers which are cheaper, more basic devices, generally with a simple light bulb that lights up when current flows.

An important application is the continuity test of a bundle of wires so as to find the two ends belonging to a particular one of these wires; there will be a negligible resistance between the "right" ends, and only between the "right" ends.

This test is performed just after the hardware soldering and configuration has been completed. This test aims at finding any electrical open paths in the circuit after the soldering. Many a times, the electrical continuity in the circuit is lost due to improper soldering, wrong and rough handling of the PCB, improper usage of the soldering iron, component failures and presence of bugs in the circuit diagram. We use a multi meter to perform this test. We keep the multi meter in buzzer mode and connect the ground terminal of the multi meter to the ground. We connect both the terminals across the path that needs to be checked. If there is continuation then you will hear the beep sound.

7.2 POWER ON TEST

This test is performed to check whether the voltage at different terminals is according to the requirement or not. We take a multi meter and put it in voltage mode. Remember that this test is performed without microcontroller. Firstly, we check the output of the transformer, whether we get the required 12 v AC voltage.

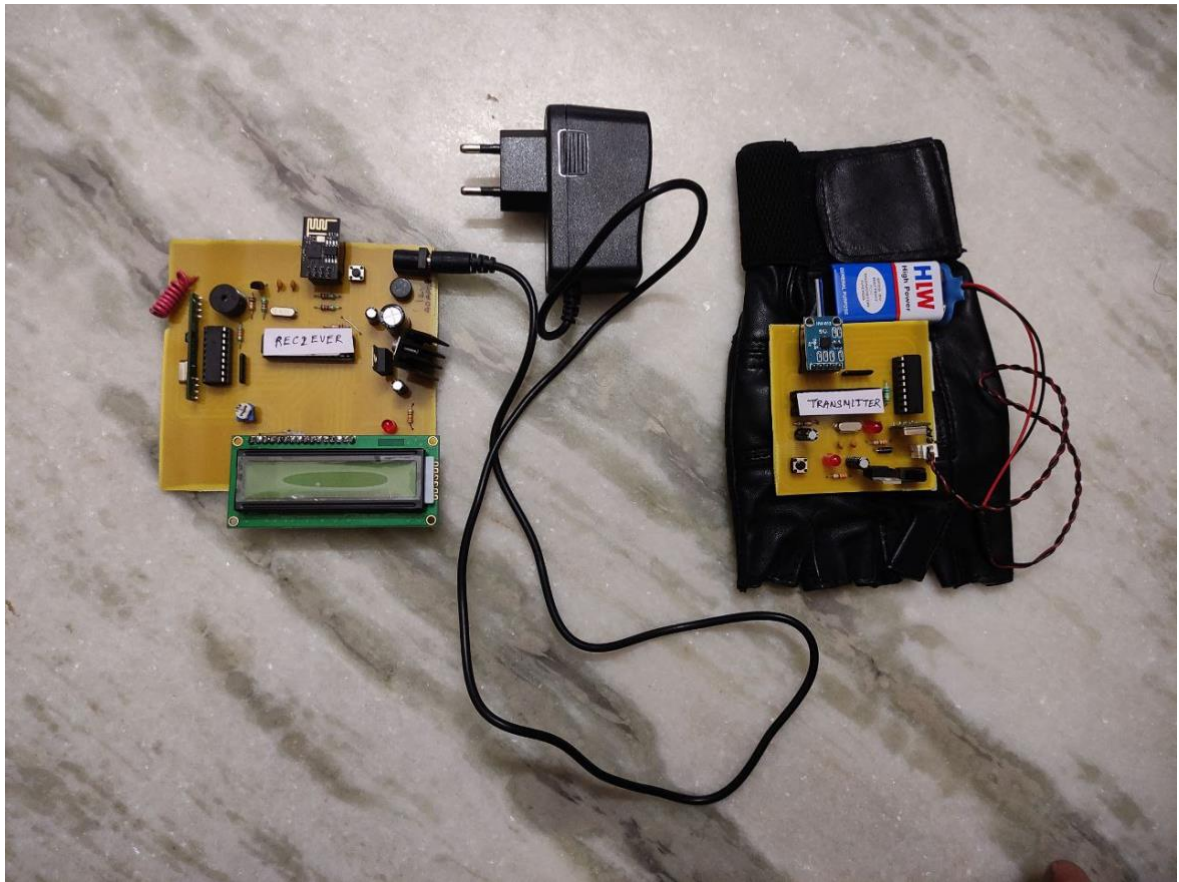
Then we apply this voltage to the power supply circuit. Note that we do this test without microcontroller because if there is any excessive voltage, this may lead to damaging the controller. We check for the input to the voltage regulator i.e., are we getting an input of 12v and an output of 5v. This 5v output is given to the microcontrollers' 40th pin. Hence we check for the voltage level at 40th pin. Similarly, we check for the other terminals for the required voltage. In this way we can assure that the voltage at all the terminals is as per the requirement.

8. OUTCOMES

- ◆ This project is useful for paralysed patients and in some cases even helps speech impaired patients too.
- ◆ It is clear that further research in the areas of feature extraction, classification methods, and gesture representation are required to realize the ultimate goal of

humans interfacing with machines on their own natural terms.

9. RESULT



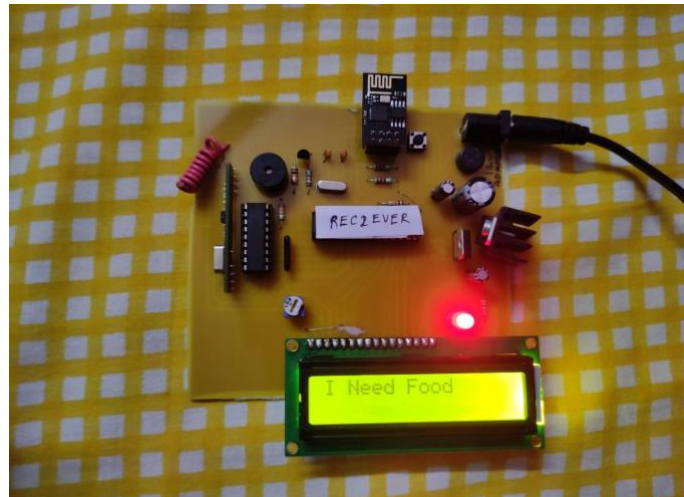
This is our final circuit connection. There latest message has now been shown on the LCD panel by the help of micro controller.

Based on the movement of the hand, the following messages are displayed on the LCD panel.

- 1) If there is **no movement** in the hand, then the message displayed will be **no input**.



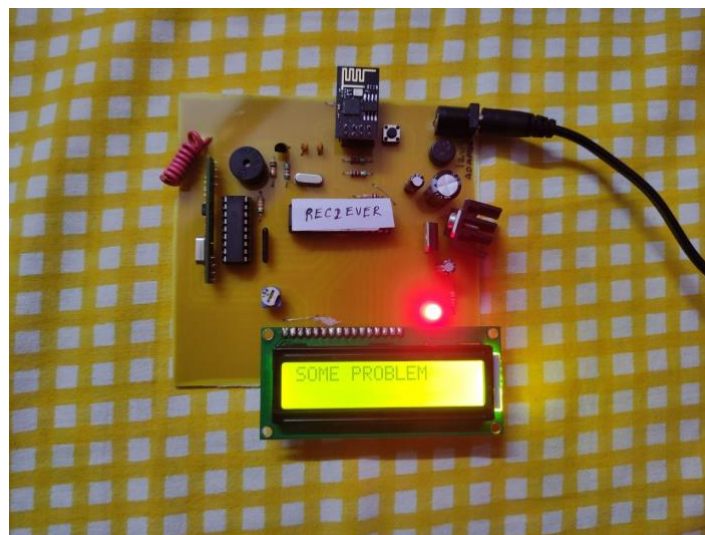
- 2) If there is a movement hand towards **DOWN**, then the message displayed will be **I NEED FOOD**



- 3) If there is a movement hand towards **UP**, then the message displayed will be **I NEED WATER**



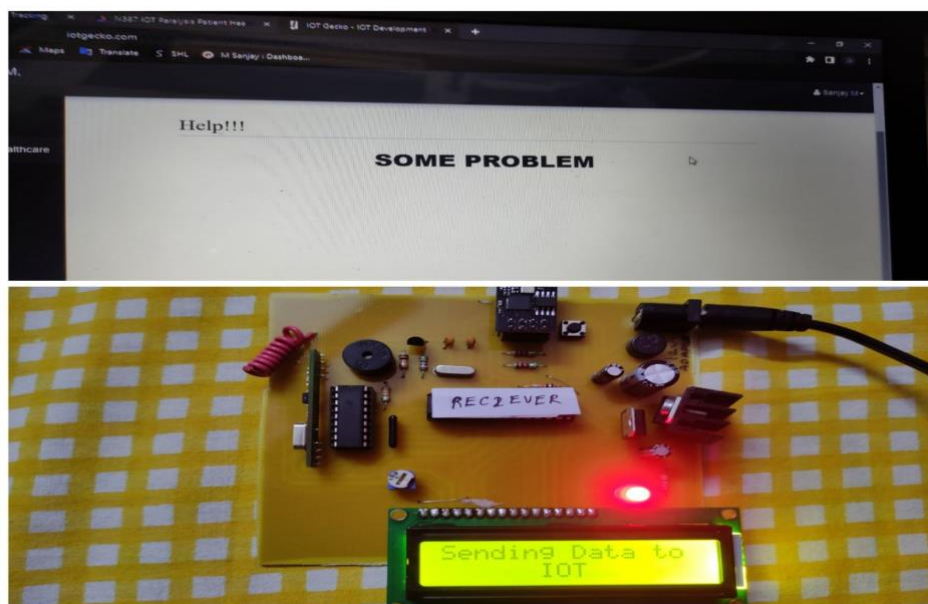
- 4) If there is a movement of hand **RIGHT**, then the message displayed will be **SOME PROBLEM**.



- 5) If there is a movement of hand **LEFT**, then the message displayed will be **CALL ATTENDANT**



The messages will be sent to IOT platform, where the attendant can see the messages.



10. CONCLUSION

This system is really helpful for paralysed patients. When they need help then they can ask by using some movements they can also survive in this world like normal people by using this movement detection. This system is reliable and cheap and less weight. so they can buy without debt. This system will make paralysed patients to achieve an independent mobility. This is not a trivial task just because it varies from person to person in its nature and type. Therefore, different methods are essential to support these people, and it is our duty, as future engineers, to develop new technologies to help paralysed patients.

11. REFERENCES

- www.atmel.com
- www.electronicworkshop.com
- www.howstuffworks.com
- <https://medlineplus.gov/paralysis.html>
- <https://www.webmd.com/brain/paralysis-types>
- www.nevonprojects.com
- www.researchgate.in
- www.wikipedia.com/paralysis