A Project report on

"SIMPLE PHONE :CALL & SMS USING GSM MODULE & ARDUINO"

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTROINCS & COMMUNICATION ENGINEERING

T. VENKATA SAI	**21BC1A0432**
K. MANIKANTA REDDY	**21BC1A0413**
S. DEVENDAR REDDY	**22BC5A0405**
S. GANGADHAR	**21BC1A0423**
A.S. LOKESH	**21BC1A0404**

Under the esteemed guidance of

Sri. K. TULASI KRISHNA, MTech (Ph.D.)

Assistant Professor, Dept. Of ECE

Submitted to

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

KANDULA OBUL REDDY MEMORIAL COLLEGE OF ENGINEERING

(Affiliated to JNTUA, Anantapur and Approved by AICTE, NEW DELHI)

KRISHNAPURAM(V), KADAPA-516003

2O24-2025

KANDULA OBUL REDDY MEMORIAL COLLEGE OF ENGINEERING

(Affiliated to JNTUA, Anantapur)

Kadapa-516003, A.P

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "Simple phone: Call & SMS Using GSM Module & Arduino" is a Bonafide record submitted by

Name of the candidate	**Reg no.**
A.S. LOKESH	**21BC1A0404**
T. VENKATA SAI	**21BC1A0432**
K.MANIKANTA REDDY	**21BC1A0413**
S. DEVENDAR REDDY	**22BC5A0405**
S. GANGADHAR	**21BC1A0423**

In partial fulfilment for the award of Degree of Bachelor of Technology in "Electronics and Communication Engineering" for the year 2024 – 2025.The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

PROJECT SUPERVISOR	**H.O.D.**
(Sri. K. TULASI KRISHNA)	**(Smt. C.Chandrakala)**

External Viva – voce Exam held on date

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

An endeavour of a long period can be successful only with the advice of many wellwishers. We take this opportunity to express our deep gratitude and appreciation to all those who encouraged me for successful completion of the project work.

Our heartfelt thanks to our Guide, **Sri. K. TULASI KRISHNA, Assistant Professor** in E.C.E. Department, Kandula Obul Reddy Memorial College of Engineering, Kadapa for his valuable guidance and suggestions in analyzing and testing throughout the period, till the end of project work completion.

Our special thanks to **Smt.C. CHANDRAKALA, HOD of Electronics and Communication Engineering**, Kandula Obul Reddy Memorial College of Engineering, Kadapa, for her timely suggestions and help in spite of her busy schedule.

We wish to express our sincere gratitude to **Smt. G. LAVANYA, Principal** of Kandula Obul Reddy Memorial College of Engineering, Kadapa, for her consistent help and encouragement to complete the project work.

We are very much thankful to **Sri. S. VENKATA SIVA REDDY, Honourable Chairman and Smt. S. LIKITHA REDDY, Correspondent** of Kandula Group of institutions, Kadapa for their help in providing good facilities in our college.

We wish to convey our gratitude and express our sincere thanks to all Project Committee members for their support and co-operation rendered for successful submission of our Project Work.

Finally, we would like to express our sincere thanks to faculty members of E.C.E department, lab technicians, and friends, one and all who has helped us complete the Project Work successfully.

As a gesture of respect for our family members and for the support we received from them, we dedicate this work to them.

ABSTRACT

In this project, we will learn how to make a Call & SMS using GSM Module & Arduino. This is a Simple Homemade Phone using GSM Module and Arduino. This simple phone is capable of calling to another number as well as receiving an incoming call. Similarly, it can also be used to send an SMS as well as read a received SMS.

So, we have interfaced GSM Module SIM800/SIM900 with Arduino UNO Board. A 16×2 LCD is used for displaying the operations that are happening like displaying signal strength, carrier name, incoming or outgoing call, SMS sent received status, and also time elapsed. The 4×4 Keypad is used to enter the number or type an SMS text using the alphanumeric keyboard. The mic is used to transmit the spoken sound and a speaker is used for ringing and listening to incoming calls/voice.

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGENO
CHAPTER 1	**INTRODUCTION**	**1-5**
1.1	OBJECTIVE OF THE PROJECT	2
1.2	RELATED WORK	3
1.3	EXISTING SYSTEM	4
1.4	PROPOSED SYSTEM	4-5
1.5	TOOLS REQUIRED	5
1.5.1	HARDWARE TOOLS	5
1.5.2	SOFTWARE TOOLS	5
CHAPTER 2	**LITERATURE SURVEY**	**6-7**
CHAPTER 3	**DOMAIN OF THE PROJECT**	**8-24**
3.1	DESIGN & IMPLEMENTATION	9
3.2	BLOCK DIAGRAM	10
3.3	DISCRIPTION OF THE BLOCK DIAGRAM	10
3.4	AVR MICROCONTROLLER	10
3.4.1	ARUDINO	10-12
3.4.2	HARDWARE	12-16
3.5	FUNCTION MODULES	17
3.5.1	POWER SUPPLY	17-18
3.5.2	MIC	18-19
3.5.3	SPEAKER	19

3.5.5	LIQUID CRYSTAL DISPLAY	20-21
3.5.6	PIN FUNCTIONS	21
3.5.7	SIM CARD	22-23
3.5.8	GSM TECHNOLOGY	23-24
CHAPTER 4	**SOFTWARE REQUIREMENTS**	**25-30**
4.1	SOFTWARE MODULE DISCRIPTION	26
4.1.1	ARDUINO INSTALLATION	26-29
4.2	ADVANTAGES & APPLICATIONS	29
4.2.1	ADVANTAGES	29
4.2.2	APPLICATIONS	29-30
CHAPTER 5	**RESULT**	**31-33**
5.1	RESULT	32-33
CHAPTER 6	**CONCLUSION & FUTURE SCOPE**	**34-35**
6.1	CONCLUSION	35
6.2	FUTURE SCOPE	35
Appendix		36-63
References		64

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGENO
Fig 3.1	EXPERIMENTAL IMPLEMENTATION OF PROJECT	9
Fig 3.2	BLOCK DIAGRAM	10
Fig 3.4.1	ARDUINO	11
Fig 3.4.2	HARDWARE	12
Fig 3.5.1	POWER SUPPLY	17
Fig 3.5.2	MIC	18
Fig 3.5.3	SPEAKER	19
Fig 3.5.4	KEY BOARD	20
Fig 3.5.5	LCD	21
Fig3.5.6	PIN MAPPING	21
Fig 3.5.7	SIM CARD	22
Fig 3.5.8	GSM TECHNOLOGY	23
Fig 5.1	SIMPLE PHONE :CALL & SMS USING GSM MODULE AND ARDUINO	32

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Objectives of project:

Global System for Mobile Communications (GSM) is widely used standard drawn by European Telecommunications Standards Institute (ETSI) for Cellular Networks used in Mobile Phones. A GSM module is a specialized type of modem which accepts a Subscriber Identity Module (SIM) card and operates over a subscription to a mobile operator just like a mobile phone. From a mobile network operator perspective a GSM modem looks just like a mobile phone. In this project SIM900 GSM module is used. The SIM900 is a complete Quadband GSM solution in a SMT module which can be embedded in the customer applications. Featuring an industry-standard interface, the SIM900 delivers GSM 850/900/1800/1900 MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption.

The Arduino IDE is also open source and anybody can contribute their libraries to the Arduino. The Arduino board has all the required circuitry to get the built-in AVR microcontroller running. The output or inputs can be taken from the boards or given to the board using convenient connectors. Both digital and analog inputs and outputs are available in all Arduino boards. The Arduino boards can also communicate with other devices using standard communication ports like USART, IIC, and USB etc. The most impressive thing is that most of the Arduino boards are bread-board compatible.

1.2 RELATED WORK :

Several studies and projects have explored the integration of GSM Modules with Arduino microcontrollers to implement basic communication functionalities such as phone calls and SMS. This section outlines key developments and prior work in this domain.

GSM-Based Home Automation Systems :

In many home automation projects, GSM modules (like SIM800L or SIM900A) have been used to send and receive SMS messages for controlling devices remotely. For instance, Patel and Shah (2017) implemented a GSM-based system that allowed users to turn appliances on and off by sending SMS commands to an Arduino-controlled system. This work demonstrated the reliability of GSM for basic remote control operations.

Security and Alert Systems :

GSM modules are commonly employed in security systems for sending alerts via SMS or initiating calls in emergencies. A study by Rani and Reddy (2018) developed a GSM-based security system that sends a message to the owner when motion is detected. The simplicity **and cost-** effectiveness of GSM made it a preferred solution in areas with limited internet connectivity.

Arduino and SIM800L Integration :

Numerous tutorials and DIY projects document the process of integrating the SIM800L GSM module with Arduino. These resources typically focus on initiating a call or sending/receiving SMS using AT commands. The practical insights provided in these works form a solid foundation for beginners to develop GSM-based communication systems.

1.3 EXISTING SYSTEM :

An existing system for simple phone call and SMS functionality using a GSM module and Arduino is typically a basic communication setup that enables sending/receiving SMS and making/receiving phone calls via the cellular network. Here’s a breakdown of how such a system works:

1. Components Used:

- Arduino UNO/Nano/Other board
- GSM Module (e.g., SIM800L, SIM900A, etc.)
- SIM Card (with an active mobile network plan)
- Power supply (regulated 5V or 3.7–4.2V for some GSM **modules)**

2. Features of the System:

- Make a Phone Call: Use AT commands to dial a number.
- Receive a Call: Detect an incoming call and optionally pick it up.
- Send SMS: Send predefined or dynamic messages to a given

  phone number.
- Receive SMS: Read messages from the inbox and act accordingly.

1.4 PROPOSED SYSTEM :

Here’s a simple proposed system for making phone calls and sending/receiving SMS using a GSM module (like SIM800L or SIM900) and an Arduino. This is useful for remote monitoring, basic communication devices, or home automation.

1. Make a Phone Call

Arduino sends an AT command to the GSM module to dial a specific number. User can also use a button press or sensor trigger to initiate the call.

2. Send SMS

Arduino sends AT command with the target phone number and message content.

3. Receive SMS

GSM module listens for incoming SMS.

Arduino reads the content and can take action accordingly (like turning on a relay).

1.5 TOOLS REQUIRED :

1.5.1 HARDWARE TOOLS

- REGULATOR POWER SUPPLY
- LCD DISPLAY
- SPEAKER
- MIC
- KEY BOARD
- SIM CARD
- GSM TECHNOLOGY

1.5.2 SOFTWARE TOOLS:

- ARDUINO UNO
- PROTEAUS

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

Prof. Dr. Sandeep M. Chaware et al. presents Garbage Monitoring system, which monitors the garbage bins and informs about the level of garbage collected in the garbage bins via a web page. Fig. 1 shows the System Architecture, in which system uses ultrasonic sensors placed over the bins to detect the garbage level and compare it with the garbage bins depth. The proposed system uses Arduino family microcontroller (The LPC2131/32/34//38 microcontrollers are based on a 16/32-bit ARM7TDMI-S CPU with real-time emulation), LCD screen, Wi-Fi modem(The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interface) for sending data and a buzzer, GSM (used to send message to the garbage depot if the Garbage Can exceeds the set threshold level) Ultrasonic Sensor. (Sensor sends out a high-frequency sound pulse and then times how long it takes for the echo of the sound to reflect back).

SUMMARY:

A GSM module with Arduino allows basic mobile communication, enabling the Arduino to send/receive SMS and make/receive phone calls through a SIM card and cellular network.

CHAPTER 3

DOMAIN OF THE PROJECT

CHAPTER 3

DOMAIN OF THE PROJECT

3.1 Design and implementation:

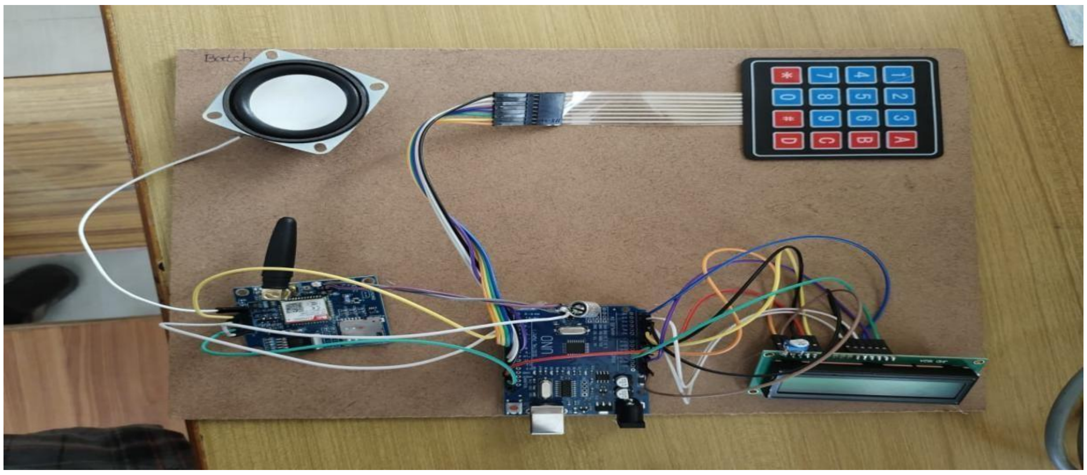


Fig 3.1 Experimental implementation of project

In modern embedded systems, wireless communication is crucial for remote monitoring and control. GSM (Global System for Mobile Communication) modules can interface with microcontrollers like Arduino to enable basic communication such as phone calls and SMS without internet access. This project focuses on implementing such a system for simple mobile communication.

3.2 Block diagram:

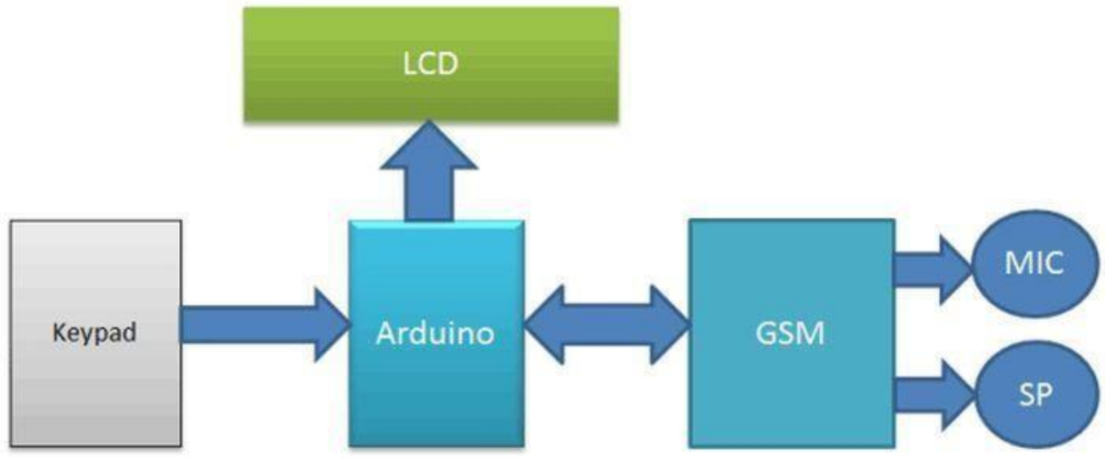


Figure 3.2: Block diagram of the project

3.3 Description of the block diagram:

The ARDUNIO ships as a bare circuit board with standard connections Keypad, gsm, lcd, mic and speaker with power supply.

3.4 AVR MICROCONTROLLER

3.4.1 ARDUNIO:

Arduino is a computer hardware and software company, project, and user community that designs and manufactures microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (*shields*) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

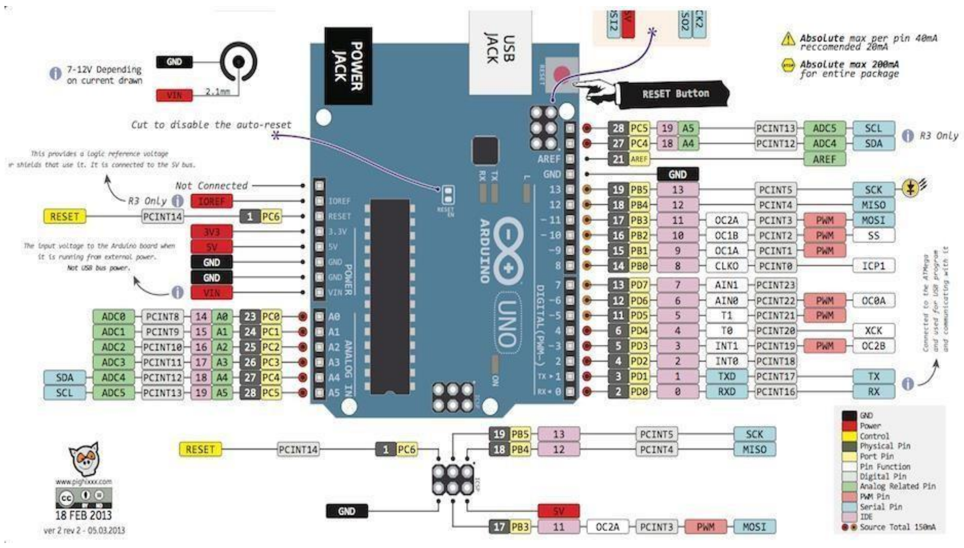


Fig 3.4.1 Arduino uno

History:

The origin of the Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy.[2] At that time, the students used a BASIC Stamp microcontroller at a cost of $100, a considerable expense for many students. In 2004, Colombian student Hernando Barragán created the development platform *Wiring* as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the Processing language. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller.[4]

3.4.2 Hardware:

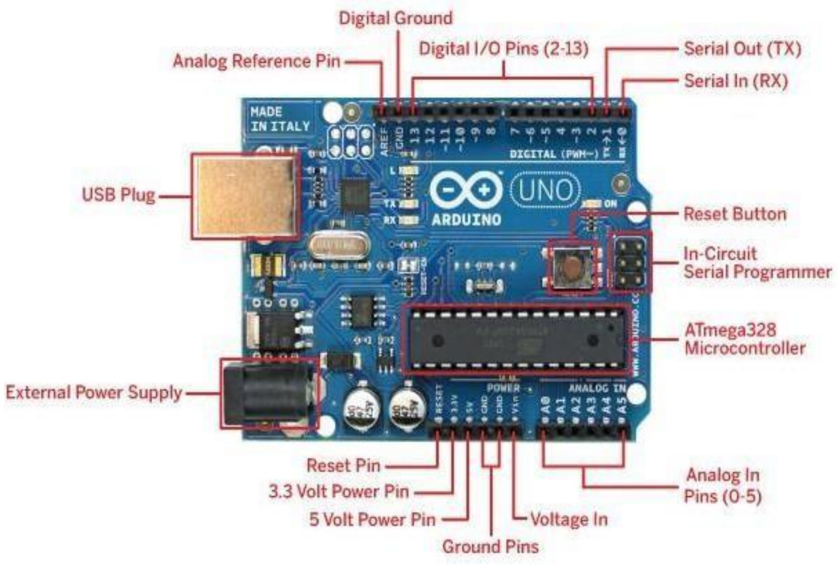


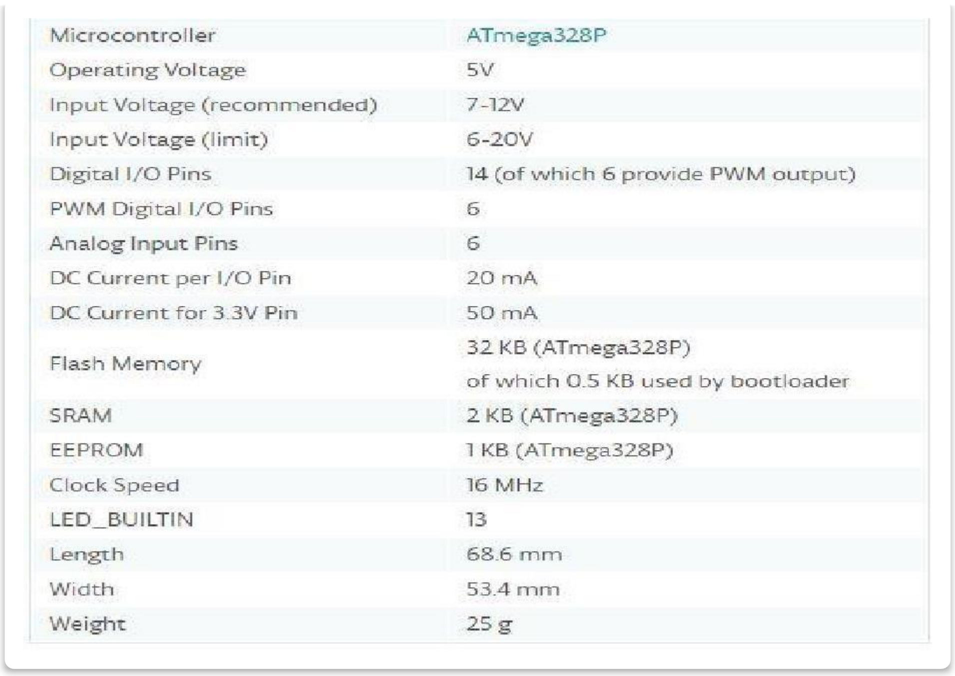
Fig3.4.2 Hardware

Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available. The source code for the IDE is released under the GNU General Public License, version 2.[8] Nevertheless, an official Bill of Materials of Arduino boards has never been released by Arduino staff.

An Arduino board consists of an Atmel 8-, 16- or 32-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560), but other makers' microcontrollers have been used since 2015. The boards use single-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed *shields*. Multiple, and possibly stacked shields may be individually addressable via an I²C serial bus. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator. Some designs, such as the LilyPad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions.

Applications:

- Xoscillo, an open-source oscilloscope[48]
- Arduinome, a MIDI controller device that mimics the Monome
- OBDuino, a trip computer that uses the on-board diagnostics interface found in most modern cars
- Ardupilot, drone software and hardware
- Gameduino, an Arduino shield to create retro 2D video games[49]
- ArduinoPhone, a do-it-yourself cellphone[50][51]
- Water quality testing platform[52]
- Automatic titration system based on Arduino and stepper motor[53]
- Low cost data glove for virtual reality applications[54]
- Impedance sensor system to detect bovine milk adulteration[55]
- Homemade CNC using Arduino and DC motors with close loop control by Homofaciens[56]
- DC motor control using Arduino
- H-Bridge[Technical specs



Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Programming:

The Arduino/Genuino Uno can be programmed with the (Arduino Software (IDE)). Select "Arduino/Genuino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

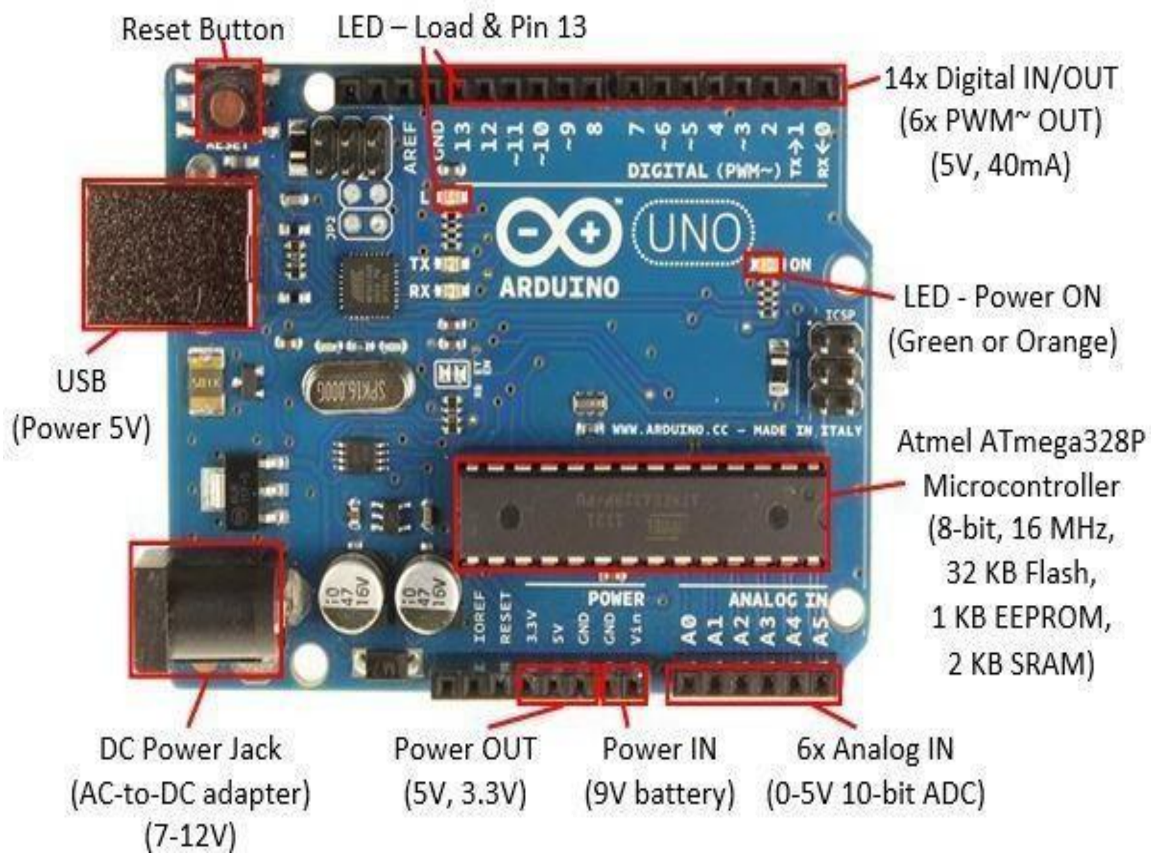
The ATmega328 on the Arduino/Genuino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

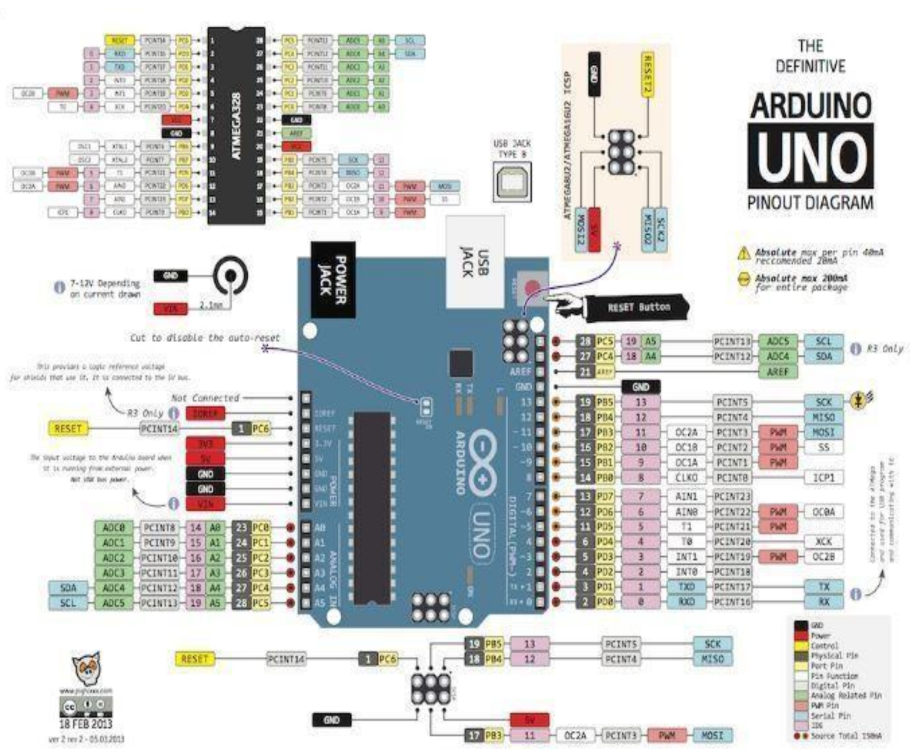
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

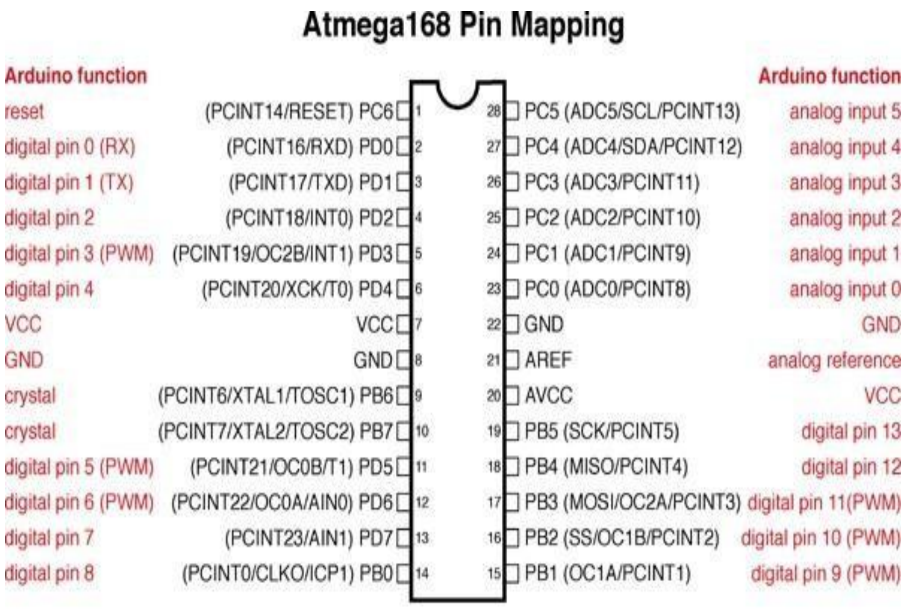
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then reseing the 8U2.

- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.
- You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.


Reset Button
LED – Load & Pin 13
14x Digital IN/OUT (6x PWM~ OUT) (5V, 40mA)
LED - Power ON (Green or Orange)
Atmel ATmega328P Microcontroller (8-bit, 16 MHz, 32 KB Flash, 1 KB EEPROM, 2 KB SRAM)
USB (Power 5V)
DC Power Jack (AC-to-DC adapter) (7-12V)
Power OUT (5V, 3.3V)
Power IN (9V battery)
6x Analog IN (0-5V 10-bit ADC)


THE DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
Cut to disable the auto-reset
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
18 FEB 2013



Atmega168 Pin Mapping

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

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

3.5 FUNCTION MODULES :

3.5.1 POWER SUPPLY:

The input to the circuit is applied from the regulated power supply. The a.c. input i.e., 230V from the mains supply is step down by the transformer to 12V and is fed to a rectifier. The output obtained from the rectifier is a pulsating d.c voltage. So in order to get a pure d.c voltage, the output voltage from the rectifier is fed to a filter to remove any a.c components present even after rectification. Now, this voltage is given to a voltage regulator to obtain a pure constant dc voltage. The block diagram of regulated power supply is shown in the figure 3.2

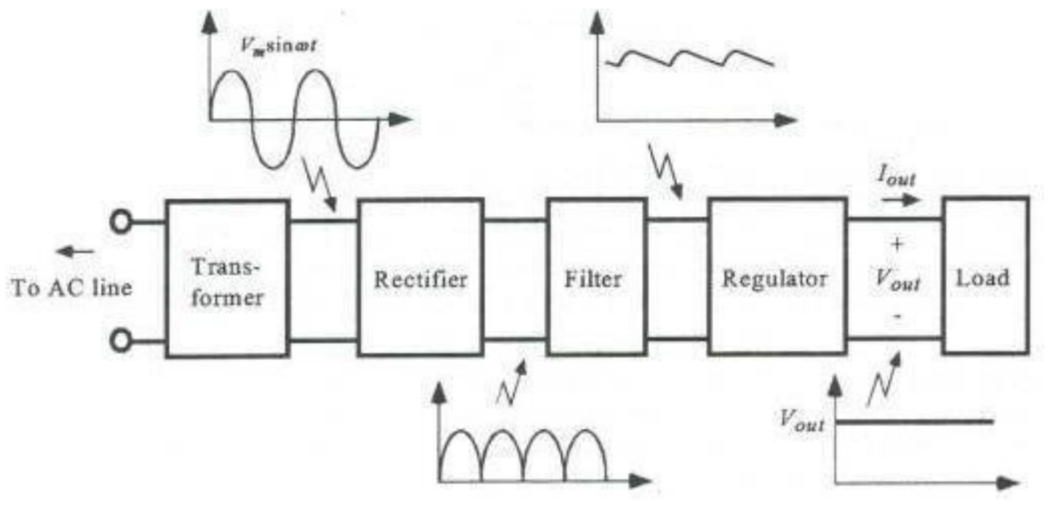


Fig 3.5.1components of power supply

Transformer:

Usually, DC voltages are required to operate various electronic equipment and these voltages are 5V, 9V or 12V. But these voltages cannot be obtained directly. Thus the a.c input available at the mains supply i.e., 230V is to be brought down to the required voltage level. This is done by a transformer. Thus, a step down transformer is employed to decrease the voltage to a required level.

Rectifier:

The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification.

Filter:

Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant.

Voltage regulator:

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power supply of 5V and 12V are required. In order to obtain these voltage levels, 7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers 05, 12 represent the required output voltage levels.

3.5.2 MIC:

Fig 3.5.2 Mic.

Microphones (or mics) are transducers that convert sound waves into electrical signals, enabling their use in a vast array of applications. They are essential for capturing and transmitting audio, playing a crucial role in communications, recording, and sound amplification.

3.5.3 SPEAKER:

Fig 3.5.3 Speaker

If you're building or exploring a talking plant with a voice module, you're probably working on a fun electronics or robotics project where a plant can "talk" using a speaker. Here's a basic overview of what you'd need and how it could work: Components You'd Use

1. Speaker – Small speaker module to output sound.
2. Voice Module – Such as:

DF Player Mini (plays MP3s from an SD card).

ISD1820 (for recording and playing short audio clips).

3. Microcontroller – Arduino, ESP32, Raspberry Pi, etc.
4. Sensors (optional) – To detect interaction (e.g., touch sensor, soil moisture sensor, light sensor).
5. Power Supply – Battery or USB power.
6. Amplifier Module (optional) – If your speaker needs extra power.

3.5.4 KEY BOAED:

Fig 3.5.4 Key Board

Depends on the user's needs and preferences while virtual keyboards on smartphones are generally adequate for most users, some find physical keyboards more efficient for tasks requiring frequent or complex typing.

3.5.5 LIQUID CRYSTAL DISPLAY(LCD):

LCD stands for **L**iquid **C**rystal **D**isplay. LCD is finding wide spread use replacing LEDs (seven segment LEDs or other multi segment LEDs) because of the following reasons:

1. The declining prices of LCDs.
2. The ability to display numbers, characters and graphics. This is in contrast to
3. LEDs, which are limited to numbers and a few characters.
4. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU
5. of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU to keep displaying the data.
6. Ease of programming for characters and graphics.
7. The ability to display numbers, characters and graphics. This is in contras to LEDs, which are limited to numbers and a few characters.

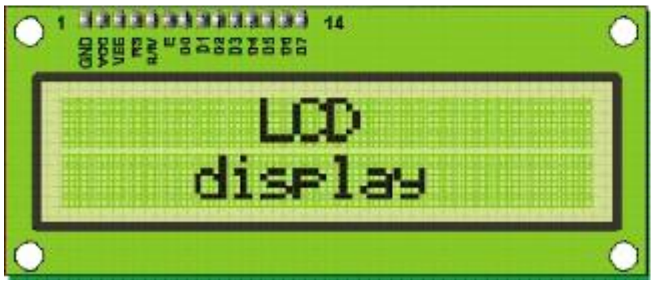


Fig 3.5.5 LCD

3.5.6 Pin diagram:

Here’s a simple and clear explanation of a basic phone system using a GSM module and Arduino, including the **pin diagram** (connections).

4. Function	**Pin Number**	**Name**	**Logic State**	**Description**
Ground	1	Vss	-	0V
Power supply	2	Vdd	-	+5V
Contrast	3	Vee	-	0 – Vdd
	4	RS	0 1	D0 – D7 are interpreted as commands D0 – D7 are interpreted as data
Control of operating	5	R/W	0 1	Write data (from controller to LCD) Read data (from LCD to controller)
	6	E	0 1	Access to LCD disabled Normal operating
	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1
	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
Data / commands	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	13	D6	0/1	Bit 6

3.5.7 SIM CARD:

Fig 3.5.7 Sim Card

A SIM card or SIM (subscriber identity module) is an integrated circuit (IC) intended to securely store an international mobile subscriber identity (IMSI) number and its related key, which are used to identify and authenticate subscribers on mobile telephone devices (such as mobile phones and laptops). SIMs are also able to store address book contacts information,[1] and may be protected using a PIN code to prevent unauthorized use.

A typical SIM card (mini-SIM with micro-SIM cutout) SIMs are always used on GSM phones; for CDMA phones, they are needed only for LTE-capable handsets. SIM cards are also used in various satellite phones, smart watches, computers, or cameras. The first SIM cards were the size of credit and bank cards; sizes were reduced several times over the years, usually keeping electrical contacts the same, to fit smaller-sized devices. SIMs are transferable between different mobile devices by removing the card itself.

A SIM card or SIM (subscriber identity module) is an integrated circuit (IC) intended to securely store an international mobile subscriber identity (IMSI) number and its related key, which are used to identify and authenticate subscribers on mobile telephone devices (such as mobile phones and laptops). SIMs are also able to store address book contacts information,[1] and may be protected using a PIN code to prevent unauthorized use.

SIM Card Sizes:

- **Full-size (1FF)**: The original size, now largely obsolete.
- **Mini-SIM (2FF)**: Common in older mobile phones.
- **Micro-SIM (3FF)**: Smaller than mini-SIM, used in many smartphones.
- **Nano-SIM (4FF)**: The smallest size, prevalent in most modern smartphones.
- **Embedded SIM (eSIM)**: A programmable SIM embedded directly into a device's hardware, eliminating the need for a physical card.

3.5.8 GSM TECHNOLOGY:

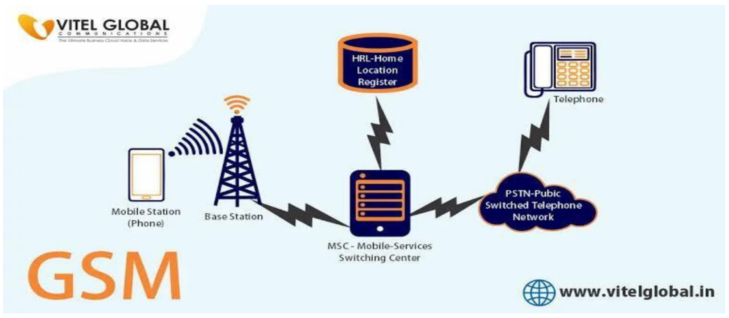

VITEL GLOBAL
HRL-Home Location Register
Telephone
Mobile Station (Phone)
Base Station
MSC - Mobile-Services Switching Center
PSTN-Pubic Switched Telephone Network
GSM
www.vitelglobal.in

Fig 3.5.8 GSM Technology

GSM, or Global System for Mobile Communications, is a digital cellular network technology that's widely used for voice and data communication. It's known for its ability to provide seamless communication across different networks and is the foundation for 2G and 3G mobile networks, although newer technologies like 4G and 5G have now become dominant.

GSM Technology

SIM Card:

GSM uses a Subscriber Identity Module (SIM) card to identify and authenticate users within the network.

2G and 3G Evolution:

GSM evolved from 2G technology (focused on voice and basic data) to 3G (with enhanced data capabilities).

Variable resistors used as potentiometers have all **three terminals** connected. This arrangement is normally used to **vary voltage**, for example to set the switching point of a circuit with a sensor, or control the volume (loudness) in an amplifier circuit. If the terminals at the ends of the track are connected across the power supply, then the wiper terminal will provide a voltage which can be varied from zero up to the maximum of the supply.

CHAPTER 4
SOFTWARE REQUIREMENTS

CHAPTER 4
SOFTWARE REQUIREMENTS

4.1 SOFTWARE MODULES DESCRIPTION

4.11 ARDUNIO INSTALLATION:

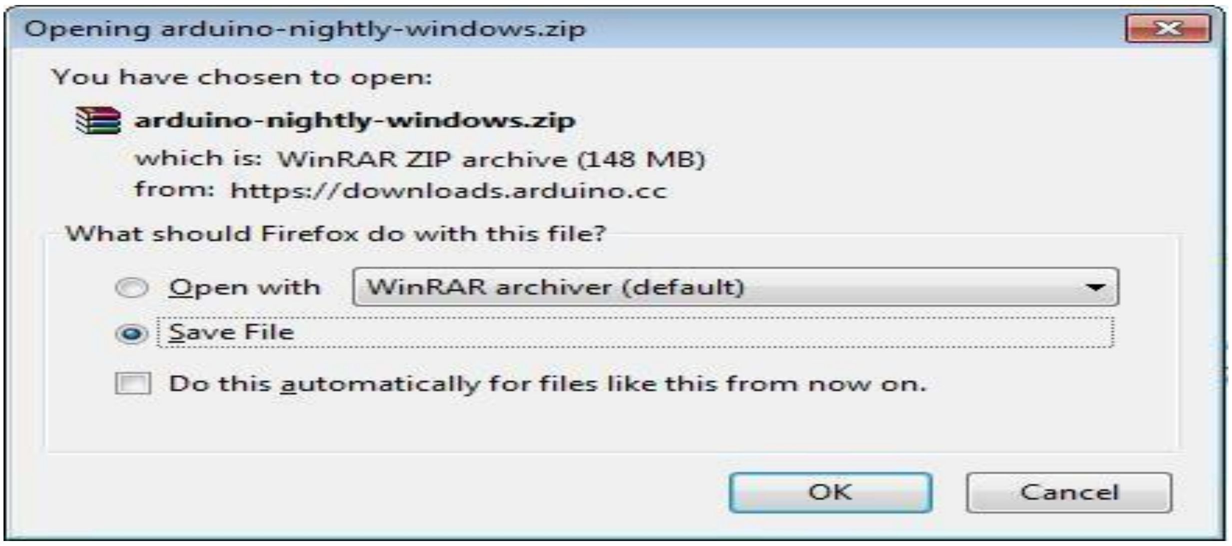
After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1: First you must have your Arduino board (you can choose your favorite board) and aUSB cable. In case you use Arduino UNO, ArduinoDuemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

Step 2: Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



Opening arduino-nightly-windows.zip

You have chosen to open:

arduino-nightly-windows.zip

which is: WinRAR ZIP archive (148 MB)

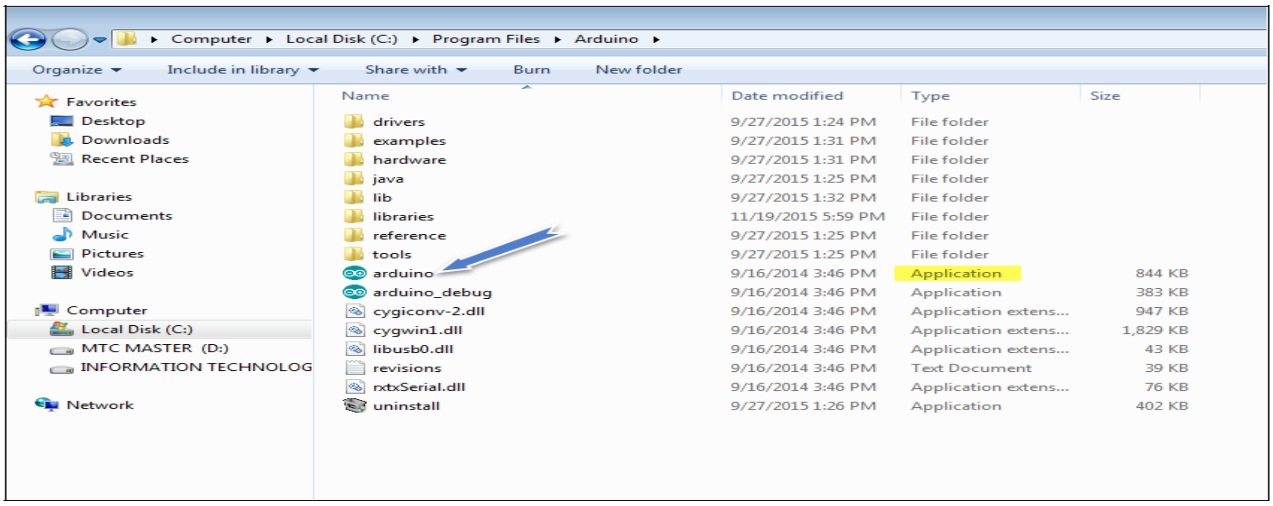
from: https://downloads.arduino.cc

What should Firefox do with this file?

Open with: WinRAR archiver (default)

[✓ Save File]

Do this automatically for files like this from now on.



Computer ▸ Local Disk (C:) ▸ Program Files ▸ Arduino ▸

Name	Date modified	Type	Size
drivers	9/27/2015 1:24 PM	File folder	
examples	9/27/2015 1:31 PM	File folder	
hardware	9/27/2015 1:31 PM	File folder	
java	9/27/2015 1:25 PM	File folder	
lib	9/27/2015 1:32 PM	File folder	
libraries	11/19/2015 5:59 PM	File folder	
reference	9/27/2015 1:25 PM	File folder	
tools	9/27/2015 1:25 PM	File folder	
arduino	9/16/2014 3:46 PM	Application	844 KB
arduino_debug	9/16/2014 3:46 PM	Application	383 KB
cygiconv-2.dll	9/16/2014 3:46 PM	Application extens...	947 KB
cygwin1.dll	9/16/2014 3:46 PM	Application extens...	1,829 KB
libusb0.dll	9/16/2014 3:46 PM	Application extens...	43 KB
revisions	9/16/2014 3:46 PM	Text Document	39 KB
rxtxSerial.dll	9/16/2014 3:46 PM	Application extens...	76 KB
uninstall	9/27/2015 1:26 PM	Application	402 KB

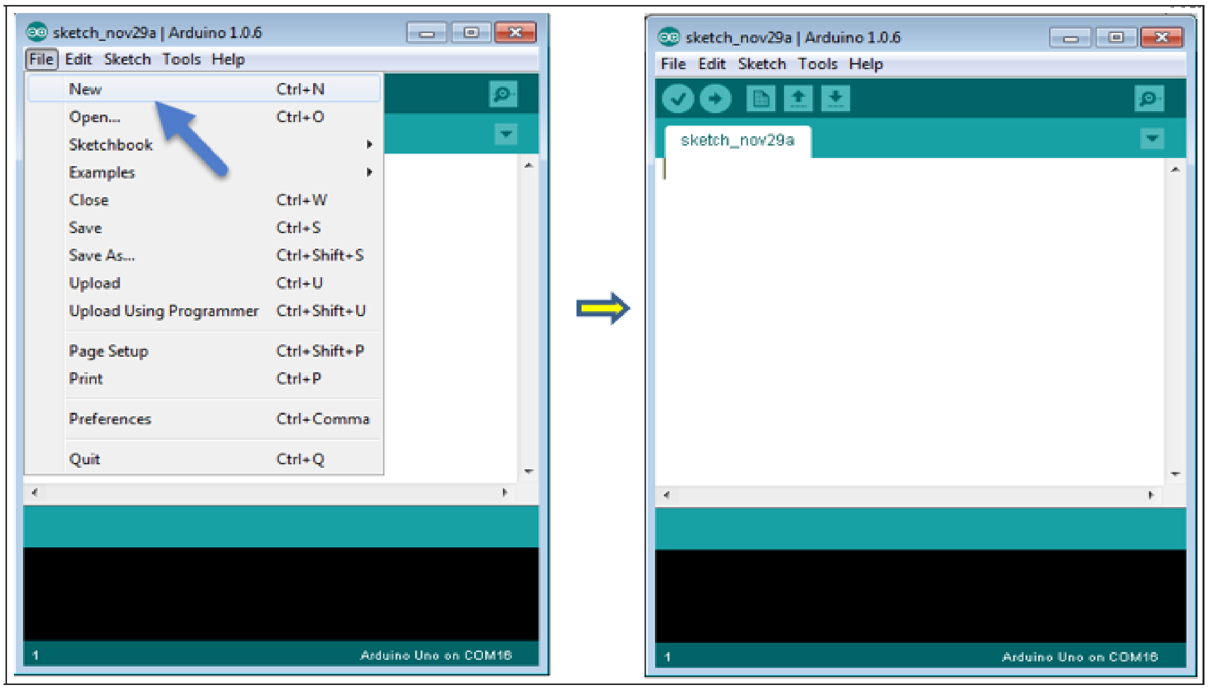
Step 3: Power up your board:

The ESP32 Uno, Mega, Duemilanove and ESP32Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an ESP32Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the ESP32 board to your computer using the USB cable. The green power LED (labeled) should glow.

Troubleshooting Tips

- Check power supply: GSM needs at least **2A current**.
- SIM card must be **unlocked** and **registered** on network.
- Use `AT+CPIN?` to check SIM status (`READY` expected).
- Use `AT+CSQ` to check signal quality.
- Use serial monitor at **9600 baud** to debug.

Step 4: Launch ESP32 IDE.After your ESP32 IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



Step 5: Open your first project.

Once the software starts, you have two options:

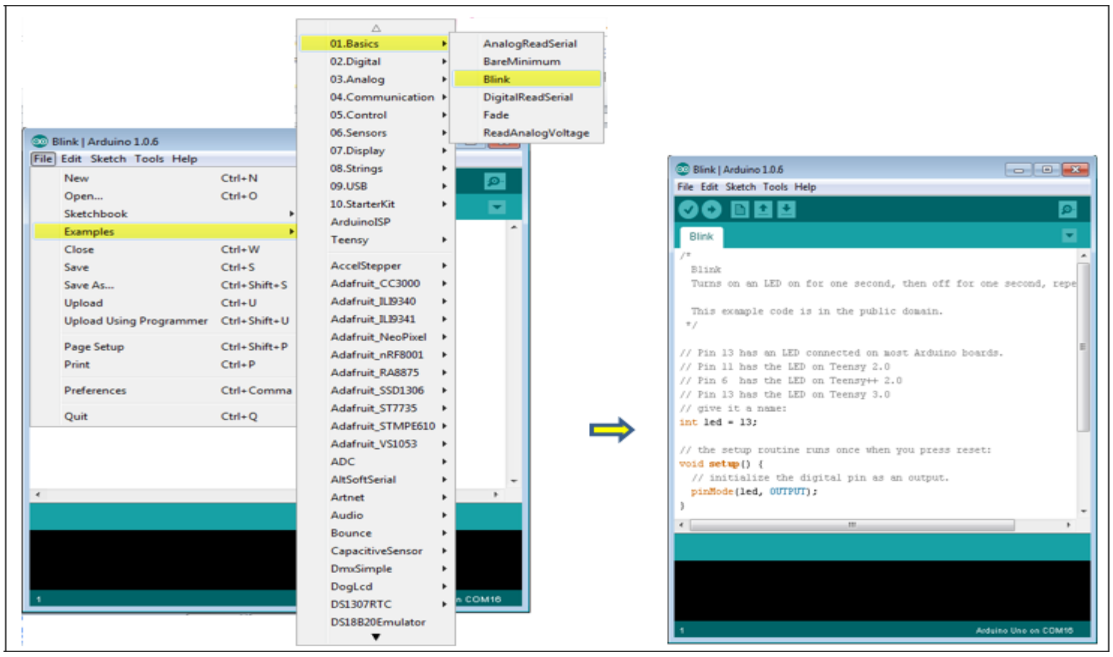
Create a new project.

Open an existing project example.

To create a new project, select File --> New

To open an existing project example, select File -> Example -> Basics -> Blink.

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

Step 6: Select your ESP32 board.

To avoid any error while uploading your program to the board, you must select the correct ESP32 board name, which matches with the board connected to your computer. Go to Tools -> Board and select your board.

4.2 ADVANTAGES & APPLICATIONS

4.2.1 ADVANTAGES:

- Low design time.
- Low production cost.
- This system is applicable for both the indoor and outdoor environment.
- Setting the destination is very easy.
- It is dynamic system.
- Less space.
- Low power consumption.

4.2.2 APPLICATIONS:

- **Emergency Alert Systems**

  - Auto-dial emergency numbers in case of fire, gas leak, or break-in.

  - Used in elderly care systems to notify caregivers.
- **Remote Control Systems**

  - Make a call to a device to trigger an action (e.g., turn ON/OFF pumps or
- machines
- **Remote Monitoring & Control**

  - Send SMS commands to check temperature, turn on/off lights, pumps, etc.

  - Receive sensor readings (e.g., humidity, power outage alerts).
- **Automated Notification Systems**

  - Send updates or alerts from IoT systems (weather station, greenhouse).

  - SMS confirmation for deliveries, bookings, or attendance.

CHAPTER 5
RESULT

5.1 RESULT:

Call & SMS using GSM Module & Arduino

In this post, we will learn how to make a Call & SMS using GSM Module & Arduino. This is a Simple Homemade Phone using GSM Module and Arduino. This simple phone is capable of calling to another number as well as receiving an incoming call. Similarly, it can also be used to send an SMS as well as read a received SMS.

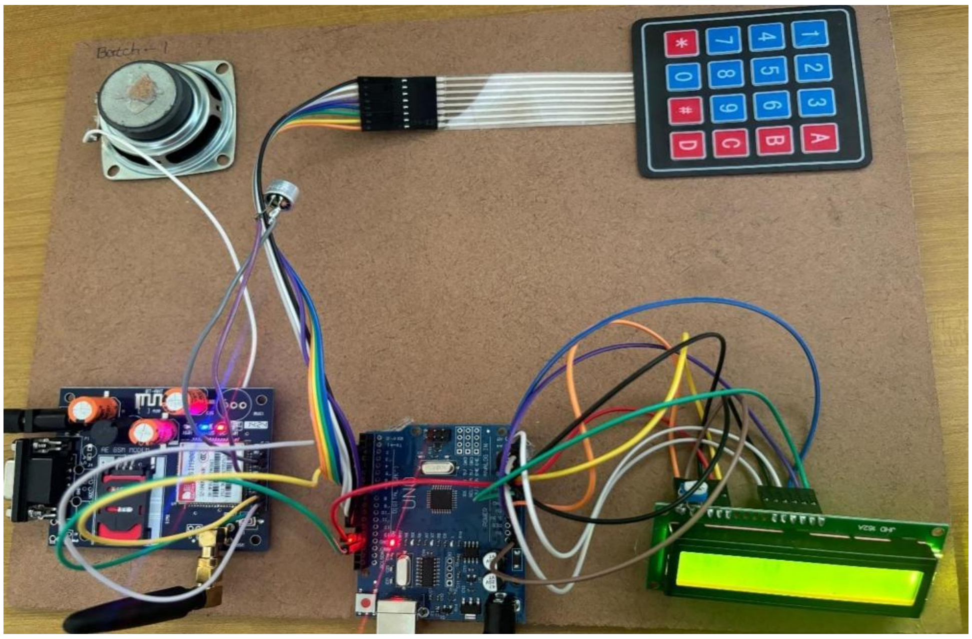


Fig5.1: Simple phone :Call & SMS Using GSM Module

So we have interfaced GSM Module SIM800/SIM900 with Arduino UNO Board.

A 16×2 LCD is used for displaying the operations that are happening like displaying signal strength, carrier name, incoming or outgoing call, SMS sent received status, and also time elapsed. The 4x4 Keypad is used to enter the number or type an SMS text using the alphanumeric keyboard. The mic is used to transmit the spoken sound and a speaker is used for ringing and listening to incoming calls/voice.

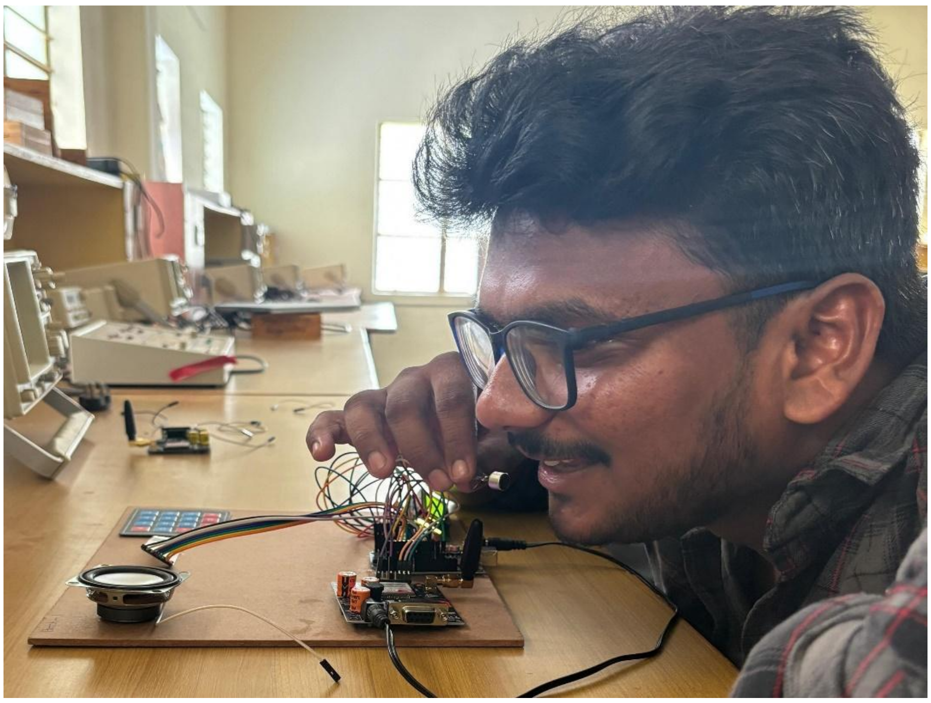


Fig 5.2 Using simple phone…

The mic is used to transmit the spoken sound and a speaker is used for ringing and listening to incoming calls/voice.

CHAPTER 6 CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION:

In this project, we successfully demonstrated how to make phone calls and send SMS messages using an Arduino and a GSM module (such as SIM800L or SIM900). By integrating the GSM module with the Arduino through serial communication, we were able to control GSM functions using AT commands. The system proved effective for basic telecommunication tasks, showing how embedded systems can be used to create simple yet powerful communication devices. This project lays the foundation for more advanced applications such as remote monitoring, IoT devices,
and emergency alert systems.

6.2 FUTURE SCOPE:

The implementation of phone call and SMS functionality using a GSM module and Arduino can be expanded in several ways:

1. Remote Monitoring Systems: Enhance the setup to monitor sensors (e.g., temperature, gas, motion) and send alerts via SMS or calls during emergencies.
2. Home Automation: Integrate with home appliances for remote control using SMS commands.
3. IoT Integration: Connect to cloud platforms via GSM for real-time data logging and monitoring in areas without Wi-Fi.
4. Security Systems: Use in low-cost security systems to alert homeowners of intrusions or hazards.

APPENDIX

```
#include <SoftwareSerial.h>
#include<String.h>
#include <LiquidCrystal.h>

SoftwareSerial mySerial(2,3); // These pins are connected to GSM module( RX, TX )

LiquidCrystal lcd(A0,A1,A2,A3,A4,A5); // These are connected to LCD pins (RS, EN,
 D4 ,D5, D6, D7 ) respectively, Vdd-5V, Vss & R/W -GND

String number= "";

String action= "WT"; //String codes: RC =Receive call, RM= Receive msg, SC= Send
 calll, SM= Send message, WT= Wait

// Receive sms Strings
String Response ="";
String sms="";
String Type;
String Caller_id;
String Text;
String SP_name="";

char character;
char quote= 0x22;

// Global Flags
bool Send_m=false;
bool sms_Receive_mode_off=true;
bool Receive_mode=false;
bool msg_Receive=false;
bool time_registered=false;
bool msg_fetched=false;
bool on_call=false;
bool start_Receive=false;
bool flag=true;

int sec,minutes; // Clock variables
long c_start;long c_time;
```

```
int i=0;

int indexOfQuotes[10];

double time_start;
double time_current;
double operational_time;

/********* Keypad Variables**********/
int r1=11;
int r2=10;
int r3=9;
int r4=8;
int c1=7;
int c2=6;
int c3=5;
int c4=4;
int colm1;
int colm2;
int colm3;
int colm4;
char value;
//
char num1;

void setup()
{
Serial.begin(9600);
Serial.println("GSM startrd");
mySerial.begin(9600);
mySerial.setTimeout(2000);
Serial.setTimeout(2000);
initilise();
lcd.begin(16, 2);
get_SP();
}
void loop()
{
Serial.println("Action: "+ action); //Reports it current mode of working
```

```
while(action=="WT") // Its wait for SMS and Calls in this loop
{
if(sms_Receive_mode_off) //So, This turns on the SMS recieve mode
{ delay(1000);
On_sms_Receive_mode();
}

if(flag)
{
Serial.println("Receive_ready");
flag=false;
print_head("Connected to:"); // Service provide name is printed on LCd
print_content(SP_name);
clear_Serial();
}


if(Receiving_on()) // FINALLY, the module is set to receive, Receive_on will beocome true
in case msg or call arrives
{
Extract_type();
}

else
{ // In case of no reciving, update the current signal strength
update_signal_strength();
get_request(); // Or, check if user pressed any button for callling or SMS
}
}

while(action=="SM") // Sending Msg action
{
Serial.println("Enter number to message");
print_head("Send SMS to");
number= Take_input(); // Take input through swith matrix
//LCD print for Send message
bool success = send_sms(number);
if(success) // If sucessful go to wait state otherwise send again
{
action="WT";
}flag=true;}
```

```
while(action=="SC") //Sending call action, similar process as above
{
print_head("Enter Call num");
Serial.println("Enter number to call");

number = Take_input();

if (valid_number()) // Check number is 10 digit long
{
print_head("Calling");
print_content(number);
delay(1000);

send_call(number);

print_head("On line with");
print_content(number);
delay(1000);
clear_Serial();
if(on_call)
{terminate_call();} // Waits here till the user is on call
}

action="WT";
flag=true;
}

while(action=="RC") // Recive call action
{
Serial.println("Press * to pick up or # to terminate");
print_head("Call from");
print_content(Caller_id);
clear_Serial();
WaitForPickup();
//incall
if(on_call) // Waits here till the user is on call
{terminate_call();}
Serial.println("Call response Recieved");
action="WT";
flag=true;}
```

```
while(action=="RM") // Recieve SMS action
{
Show_sms();
action="WT";
flag=true;
}
}

/*
* Function to get the service provider(SP) name
* Sets a Global varible: SP_name
*/

void get_SP (void)
{ bool got_it=false;  delay(1000); mySerial.println("AT"); delay(500);
print_head("Connecting GSM");
while(!( SP_name.indexOf("")>0))
{ if(GSM_operational())
{
mySerial.println("AT+COPS?"); //AT command for getting serivce provider name
mySerial.println();
}

delay(1000);

while(mySerial.available())
{
char character=mySerial.read();
SP_name.concat(character);
}

}
// Extracton process
SP_name= (SP_name.substring(SP_name.indexOf("")+1,SP_name.lastIndexOf("")));
Serial.println("Connected to: "+ SP_name);

}

// Fuciton to print current signal strength on lcd
```

```
void update_signal_strength (void)
{ String Network;
long Strength;
mySerial.println("AT+CSQ");
mySerial.println();

delay(500);
while(mySerial.available())
{
char character=mySerial.read();
Network.concat(character);
}

Network=Network.substring(Network.indexOf(':')+2,Network.indexOf(','));
Strength= Network.toInt(); // Strength Int value here

Strength=(Strength*100/31); // MAX strength= 31
lcd.setCursor(13,2);
lcd.print(int(Strength));
lcd.print('%');
}

//It recives a the char value of key pressed and stores it into
void get_request (void)
{
value=Return_Keypad_Values();
event(value);
}

// Select the apt mode as per the input
void event(char func)
{
switch (func)
{

case 'A':
action="SC"; //Send call
break;
case 'B':
action="SM"; // Send Message
break;
case 'C':
```

```
break;
default:
  action="WT"; // Wait for response
  break;
  }
  }

  /*
  * Input: (string:num,)
  * Output bool( t=sent f=unsent)
  * Function to send sms to number
  */
  bool send_sms (String number)
  {
  delay(1000);
  mySerial.println("AT");
  delay(500);
  if(GSM_operational())
  {
  mySerial.println("AT+CMGF=1");
  delay(500);
  }

  if(GSM_operational())
  {
  mySerial.print("AT+CMGS=\""); // Send the SMS number
  mySerial.print(number);
  mySerial.println("\"");

  delay(1000);
  mySerial.print("GSM bot functonal"); // SMS BODY here in case u want change
  // mySerial.print(i);
  delay(500);

  mySerial.write(0x1A);
  mySerial.write(0x0D);
  mySerial.write(0x0A);
  Serial.println("SMS sent");

  print_head("SMS Sent to");
  print_content(number);
```

```
return(true); //SMS sent succussfuly
}
return(false); // Failed attempt
}

/*
* Input: (string:num,)
* Output bool( t=sent f=failed)
* Function to send call to number
*/
bool send_call (String number)
{

mySerial.println("AT");
delay(500);

if(GSM_operational())
{
//Number dialing
Serial.println("Calling to :" +number);
print_head("Calling to");
print_content(number);
mySerial.println("ATD"+ number +";"); // AT command for dialing up the number
mySerial.println();
on_call=true;
return(true);
}
return(false);
}

// This is to switch on the messaging mode of Gsm
void On_sms_Receive_mode (void)
{
mySerial.print("ATE0");
delay(500);

if(GSM_operational())
mySerial.print("AT");
delay(500);

if(GSM_operational())
mySerial.print("AT+CMGF=1"); // Setup in msging mode
```

```
if (GSM_operational())
{
mySerial.print("AT+CNMI=2,2,0,0,0\r" ); //Prints the msg on serial as soon as it arrives
delay(500);

while(mySerial.available())
{
char response = mySerial.read();
Serial.println(response);
}

Serial.println("Receive mode On");
sms_Receive_mode_off=false; //turn it on off
}
}

/*
* Input: none
* Output: True: A response( call or sms) incoming, Or false
*
*/
bool Receiving_on (void)
{
bool Response_available=false;

if(mySerial.available())
{
while(!halt_fetch()) //In case of incoming recieve until halt_fetch() gives true
{
while(mySerial.available())
{
if(!time_registered) //Capture the time of start of message receiving
{
time_start=millis();
time_registered=true;
}
char character=mySerial.read();
Response.concat(character);
Serial.print(character); // Store as a string
}
Serial.println("Response Received"); //Looks like we got something
Response_available=true;
msg_fetched=false;
flag=true;

}
return (Response_available);
}

/*
*The function is created to halt or to indicate the end of receiving
*It does that by a timeout of 3sec or Response Text limit of 500 characters
*Input: none
*Output: Boolean, T= halt fetching F= Wait for message

*/
bool halt_fetch (void)
{
bool halt=false;

if(time_registered)
{
time_current=millis();
operational_time=time_current-time_start;
}

if(operational_time>3000 || Response.length()==200 ) // Halt condition
{
halt=true;
operational_time=0;
}
return halt;
}

/*
* It extracts the Response and caller id
* It does that by quotes position.
* Caller id is between first and second quotes
* While, Text message is after last quotes
*/
```

```
{
if(valid_input())
{
Serial.println("Valid respone");
extract();


Serial.println(Response); //In case u want to see everything incoming
Serial.println("Type: ");
Serial.print(Type);
Serial.println("Caller id : ");
Serial.println(Caller_id);
Serial.println("Text: ");
Serial.println(Text);
callORsms();
Serial.print(Caller_id);
}

time_registered=false;

Response=""; //Clear string for refresh incoming

}

/*
* Checks the validity condition,
* True: Its call or msg Resonse
* False: it is some junk
*/
bool valid_input (void)
{
bool validity;

validity=(( Response.indexOf('+') > 0) && (Response.indexOf("")>0 )); //If the reponse has
these two thing that means it is a 'real' response

if(!validity)
{
Serial.println("invalid input");

}
```

```
}

// Find the indexes of all the quotes in the stirng and sets them up in gloablevariable:
indexOfQuotes[index]
void extract(void)
{
int Length,i,index=0;

Length=Response.length();
for(i=0;i<=Length;i++)
{
if(Response.charAt(i)==quote)
{
indexOfQuotes[index]=i;
index++;
}
}

Type=Response.substring(1,indexOfQuotes[0]);
Caller_id=Response.substring(indexOfQuotes[0]+1,indexOfQuotes[1]);
Text=Response.substring(indexOfQuotes[5]+3);
Serial.println("Extracted");
}

// Determine weather the response is of call or sms
void callORsms (void)
{
if( Type.indexOf("LIP")>0) //Call string consist this( +CLIP)
{ action="RC";
Serial.println("Call from: ");}
else if(Type.indexOf("MT")>0 ) // Msg stirng consist (+CMT)
{ action="RM";
Serial.println("Message from: ");}
}

// Waits till customer press * or #
void WaitForPickup (void)
{
char key;
bool user_wait = true; //default state
while(user_wait)
```

```
key=Return_Keypad_Values();
if(key=='*') //picking up reponse
{
mySerial.println("ATA");
mySerial.println();
Serial.println("Call picked up");
print_head("Call picked up");
print_content(Caller_id);
delay(1000);
user_wait=false;
on_call=true;
}

if(key=='#') //Termination action
{
mySerial.println("ATH");
mySerial.println();
Serial.println("Call Terminated");
print_head("Call Terminated");
delay(1000);
print_content(Caller_id);
on_call=false;
user_wait=false;
}

}

}

/*
* This function is used after two user get connected on a call
* It waits '#' to terminate or 'NO CARRIER' on serial monitor
* It updates clock untill waiting
* */

void terminate_call (void)
{
char key;
bool user_wait = true; //default state
start_clock();
while(user_wait)
```

```
user_wait=check_termination();

key=Return_Keypad_Values();

if(key=='#')
{
mySerial.println("ATH"); //Termination action
mySerial.println();
Serial.println("Call Terminated");
print_head("Call Terminated");
delay(1000);
print_content(Caller_id);
user_wait=false;
}

else
{
update_clock();
}

}
on_call=false;
}

// Function to start a clock
void start_clock (void)
{
lcd.clear();
c_start=millis();
sec=0;
minutes=0;
lcd.print("On call");
}

// Function to update the value as arduino internal clock

void update_clock (void)
{
long current= millis();

if(current-c_start>1000)
```

```
sec++;
c_start=current;
}

if(sec>59)
{
minutes++;
sec=-0;
}

lcd.setCursor(0,1);

if(minutes<10)
{lcd.print('0');}
lcd.print(minutes);
lcd.print(':');
if(sec<10)
{lcd.print('0');}
lcd.print(sec);
}

// Fuction to Show sms on a LCD
void Show_sms (void)
{
print_head("SMS from");
print_content(Caller_id);
char key;


// Enhance modularity
bool user_wait = true;
while(user_wait)
{key=Return_Keypad_Values();
if(key=='*')
{
print_head(Text.substring(0,16)); // This can scroll SMS
print_content(Text.substring(16,32));
delay(2000);
print_head(Text.substring(16,32));
print_content(Text.substring(32,48));
delay(2000); //A scroll fuction can be made
Serial.println(Text);
}

if(key=='#')
{
print_head("OK");
Serial.println("MSG Terminated");
delay(500);
user_wait=false;
}

}
}

//True if starkey is pressed
bool Starkey_pressed (void)
{ char key;
key=Return_Keypad_Values();
return (key=='*');
}

//True if Hashkey is pressed
bool Hashkey_pressed (void)
{ char key;
key=Return_Keypad_Values();
return (key=='#');
}

//Check if 'NO CARRIER' is printer on Serial monitor

bool check_termination (void)
{
bool check=true;
String listen_no="";

while(mySerial.available())
{
char data= mySerial.read();
Serial.print(data);
listen_no.concat(data);
}

if(listen_no.indexOf("CAR")>0) // I check for only CAR
```

```
return check;
}

// A Fuciton to check the lenth of number calling should be 10 + ('+91' country code) =13
bool valid_number (void)
{
bool valid=false;
if(number.length()==13) // condition here
{valid=true;}
else
{print_head("Invalid input");
delay(1000);
}
return valid;
}

//Essential command to determine the state of GSM module
bool GSM_operational(void)
{
int count =0;
bool  Status=false;
mySerial.println();
while(1)
{
if (mySerial.available()>0)
{
char data = mySerial.read();
if( data == 'O' || data == 'K') //Its working properly
{
Serial.println("OK");
Status=true;
break;
}
if( data == 'E' || data == 'R' || data== 'O') // Working yet busy with some thing else
{
Serial.println("GSM not functional");
Status=false;
break;
}
}
count++;
```

```
if (count == 100)
{
Serial.println("GSM not connected"); // No reponse for AT commands
Status=false;
break;
}
}
return Status;
}

void clear_Serial (void)
{
while(mySerial.available())
{
char character=mySerial.read();
Serial.print(character);
}

}

/*************************************************************
* Keypad Firmware Ahead *
*************************************************************/

/*
* input: none
* Output: A 13 digit number
* Waits till user enter a ten Digit number
*/
String Take_input (void)
{ String num="+91";
int len=0;
int len2;
while (len <= 13)
{
len=num.length();
num1=Return_Keypad_Values();
if ((num1!='A')&&(num1!='B')&&(num1!='C')&&(num1!='a'))
{
if ((num1!='#') && (num1!='*') && (num1!='D'))
{num+=String(num1);
print_content(num);
}

else if (num1=='*')
{
num.setCharAt(len-1,'*');
print_content(num);
num.remove(len-1);

}
else if (num1=='#')
{
Serial.println(num);
break;
}
else if(num1=='D')
{
break;
}
}

}
return num;
}


void initilise()
{
pinMode(r1,OUTPUT);
pinMode(r2,OUTPUT);
pinMode(r3,OUTPUT); //use in setup
pinMode(r4,OUTPUT);
pinMode(c1,INPUT);
pinMode(c2,INPUT);
pinMode(c3,INPUT);
pinMode(c4,INPUT);
Serial.begin(9600);
digitalWrite(c1,HIGH);
digitalWrite(c2,HIGH);
digitalWrite(c3,HIGH);
digitalWrite(c4,HIGH);
}
void row1()
```

```
digitalWrite(r1,LOW);
digitalWrite(r2,HIGH);
digitalWrite(r3,HIGH);
digitalWrite(r4,HIGH);
}
void row2()
{
digitalWrite(r1,HIGH);
digitalWrite(r2,LOW);
digitalWrite(r3,HIGH);
digitalWrite(r4,HIGH);
}
void row3()
{
digitalWrite(r1,HIGH);
digitalWrite(r2,HIGH);
digitalWrite(r3,LOW);
digitalWrite(r4,HIGH);
}
void row4()
{
digitalWrite(r1,HIGH);
digitalWrite(r2,HIGH);
digitalWrite(r3,HIGH);
digitalWrite(r4,LOW);
}
void ReadRows()
{
colm1=digitalRead(c1);
colm2=digitalRead(c2);
colm3=digitalRead(c3);
colm4=digitalRead(c4);
}
char Return_Keypad_Values(void)
{
row1();
ReadRows();
delay(100);
if(colm1==LOW)
{
Serial.println("1");
delay(200);
}
else if(colm2==LOW)
{
Serial.println("2");
delay(200);
return '2';
}
else if(colm3==LOW)
{
Serial.println("3");
delay(200);
return '3';
}
else if(colm4==LOW)
{
Serial.println("A");
delay(200);
return 'A';
}

row2();
ReadRows();
delay(100);
if(colm1==LOW)
{
Serial.println("4");
delay(200);
return '4';
}
else if(colm2==LOW)
{
Serial.println("5");
delay(200);
return '5';
}
else if(colm3==LOW)
{
Serial.println("6");
delay(200);
return '6';
}
else if(colm4==LOW)
```

```
Serial.println("B");
delay(200);
return 'B';
}
row3();
ReadRows();
delay(100);
if(colm1==LOW)
{
Serial.println("7");
delay(200);
return '7';
}
else if(colm2==LOW)
{
Serial.println("8");
delay(200);
return '8';
}
else if(colm3==LOW)
{
Serial.println("9");
delay(200);
return '9';
}
else if(colm4==LOW)
{
Serial.println("C");
delay(200);
return 'C';
}
row4();
ReadRows();
delay(100);
if(colm1==LOW)
{
Serial.println("*");
delay(200);
return '*';
}
else if(colm2==LOW)
{
```

```
delay(200);
return '0';
}
else if(colm3==LOW)
{
Serial.println("#");
delay(200);
return '#';
}
else if(colm4==LOW)
{
Serial.println("D");
delay(200);
return 'D';
}
return 'a';
}

/***************************************************************
* LCD functions Ahead *
***************************************************************/

//Print out the Heading On lCD
void print_head (String str)
{ lcd.clear();
lcd.setCursor(0,0);
lcd.print(str);
}

//Print secondary content on LCD
void print_content (String str)
{
lcd.setCursor(0,1);
lcd.print(str);
}
```

REFERENCES

1. Marco Schwartz, Arduino home automation projects, Shroff Publishers, 2014.

2. Bharaka k, Ghobril M, Malek S, Khanj R, ”Low cost Arduino based smart automation” in proceedings on IEEE International conference on Computational Intellingence, Communication systems and Networks(CICSyN), Madrid, June 2013.

3. Joerg Eberspaecher, Hans-Joerg Voegel, Christian Bettstetter GSM Switching, Services and Protocols, John Wiley Publishers.

4. Rogier Noldus,Intelligent Networks for the GSM, GPRS and UMTS Network,John Wiley Publisher.