6 PROGRAMMING WITH ARDUINO

6.1 Introduction

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed referred to as a microcontroller and ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board. There are different types of Arduino boards. They are

- Arduino boards based on ATMEGA328 microcontroller.
- Arduino boards based on ATMEGA32U4 microcontroller.
- Arduino boards based on ATMEGA2560 microcontroller.
- Arduino boards based on AT91SAM3X8E microcontroller.

6.1.1 Features of Arduino

1. Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
2. We can control our board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
3. Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. We can simply use a USB cable.
4. Arduino IDE uses a simplified version of C++, making it easier to learn to program.
5. Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

6.2 Components of Arduino Board

In this section, we will learn about the different components on the Arduino board. Here the

Arduino UNO board is used because it is the most popular board in the Arduino board family. In addition, it is the best board to get started with electronics and coding. Figure 6.1 shows an Arduino UNO board. Some boards look a bit different from the one given below, but most Arduinos have majority of these components in common.

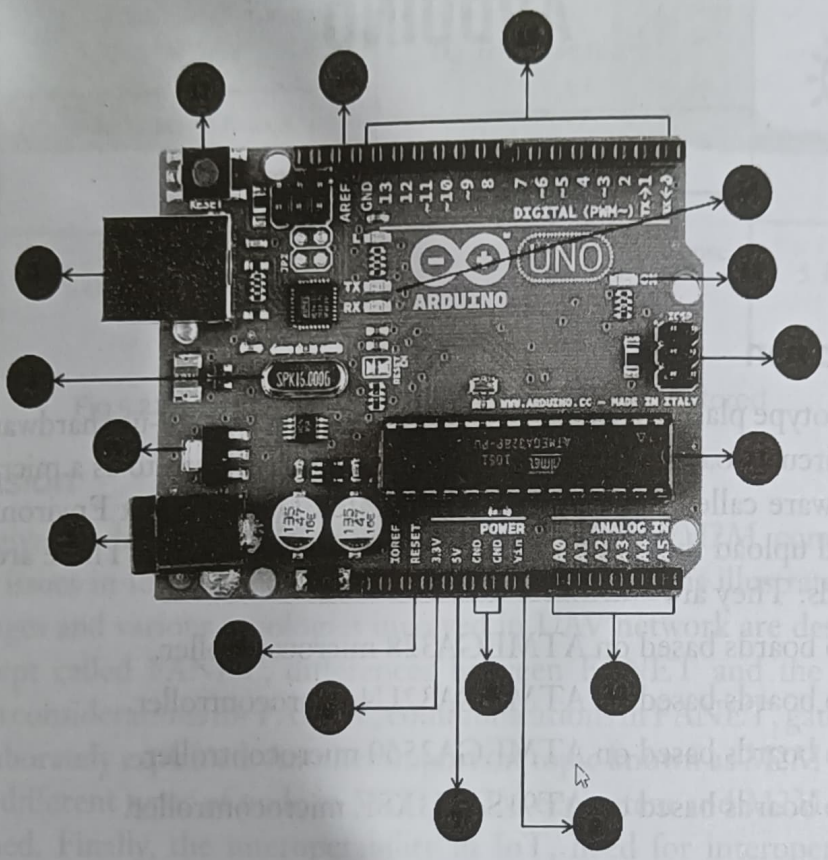

Fig.6.1: Arduino UNO Board

1. **Power USB**: Arduino board can be powered by using the USB cable from the computer. For this we need to connect the USB cable to the USB connection.
2. **Power (Barrel Jack)**: Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack.
3. **Voltage Regulator**: The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.
4. **Crystal Oscillator:** The crystal oscillator helps Arduino in dealing with time issues. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.
5. **Arduino Reset**: We can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, we can connect an external reset button to the Arduino pin labelled RESET (5).
6. **Pin (3.3)**: Supply 3.3 output volt

7. **Pin 5V:** Supply 5 output volt
8. **GND (Ground)**: There are several GND pins on the Arduino, any of which can be used to ground our circuit.
9. **Vin (9)**: This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.
10. **Analog Pins**: The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.
11. **Main Microcontroller**: Each Arduino board has its own microcontroller. We can consider it as the brain of our board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. We must know what IC our board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, we can refer to the data sheet.
12. **ICSP Pin**: ICSP is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. We are making the output device a slave to the master of the SPI bus.
13. **Power LED indicator**: This LED should light up when you plug your Arduino into a power source to indicate that our board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.
14. **TX and RX LEDs**: On the board, there will be two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First at the digital pins 0 and 1. This indicates that the pins are responsible for serial communication. Second, the TX and RX led. The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.
15. **Digital I/O**:The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.
16. **AREF**: AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

6.3 Arduino IDE

Arduino IDE is open source software that is used to program the Arduino controller board. It is based on variations of C and C++programming language. It can be downloaded from

Arduino's official Website and installed into PC. Following steps are required to set up an Arduino board.

1. Power the board by connecting it to a PC via USB cable: 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. The green power LED (labelled PWR) should glow.
2. Launch the Arduino IDE: After the Arduino IDE software is downloaded, we need to unzip the folder. Inside the folder, we can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.
3. Open the first project: Once the software starts, we have two options – Create a new project or open an existing project example. To create a new project, select File → New. To open an existing project example, select File → Example.
4. Select the Arduino board -To avoid any error while uploading our program to the board, we must select the correct Arduino board name, which matches with the board connected to your computer. Go to Tools → Board and select your board. Here, we will be selecting Arduino Uno board, but we must select the name matching the board that we are using.
5. Select the serial port - Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, we can disconnect our Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.
6. Upload the program to our board - Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar. Figure 6.2 explains various symbols on Arduino IDE toolbar.

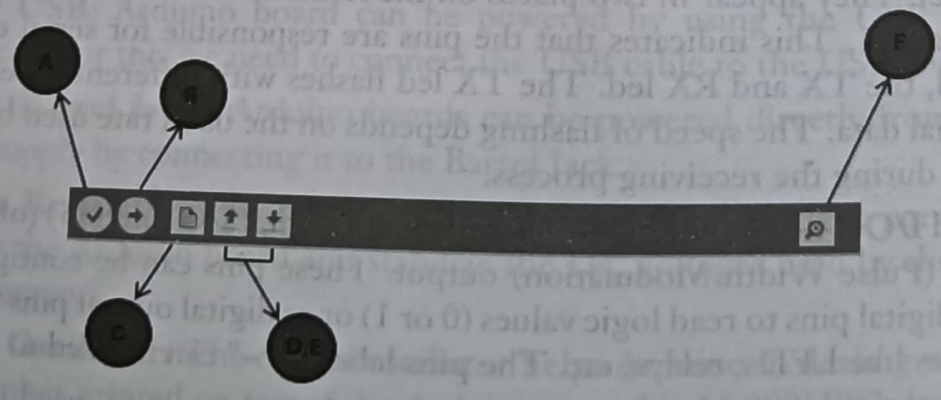

Fig.6.2: Symbols on Arduino IDE Toolbar

Program coded in Arduino IDE is called a SKETCH. Following are the explanation of various symbols used in the above diagram.

A – used to check if there is any compilation error.

B – used to upload a program to the Arduino board.

6.4.1 Program Structure

Arduino programs can be divided in three main parts: **Structure,Values (variables and constants) and Functions**. Let us start with the Structure. Values and Functions will be discussed in the coming sections. Structure consist of two main functions – setup () function and loop () function.

setup () function - The setup() function is called when a sketch starts. It is used to initialize the variables, pin modes, start using libraries, etc. The setup () function will only run once, after each power up or reset of the Arduino board.

loop () function - After creating a setup() function, which initializes and sets the initial values, the loop() function allows our program to iteratively do the specified task in the program. This is used to actively control the Arduino board.

6.4.2 Variables

6.4.10 Time

Arduino provides four different time manipulation functions. They are given below.

1. **delay ()** function - It accepts a single integer (or number) argument. This number represents the time (measured in milliseconds).
2. **delayMicroseconds ()** function - The delayMicroseconds() function accepts a single integer (or number) argument. There are a thousand microseconds in a millisecond, and a million microseconds in a second.
3. **millis ()** function- This function is used to return the number of milliseconds at the time the Arduino board begins running the current program.
4. **micros ()** function - The micros() function returns the number of microseconds from the time the Arduino board begins running the current program. This number overflows i.e. goes back to zero after approximately 70 minutes.

6.5 Function Libraries

There are several function libraries in Arduino that contains Input-Output functions, character functions, math library and trigonometric functions. We will see one by one.

6.5.1 Input-Output Functions

The pins on the Arduino board can be configured as either inputs or outputs using the **`pinMode()`** function. Arduino pins are by default configured as inputs, so they do not need to be explicitly declared as inputs with **`pinMode()`** when we are using them as inputs. Pins configured as INPUT are said to be in a high-impedance state. Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. The syntax of **`pinMode()`** function is as follows.

```
void setup () {
  pinMode(pin,mode);
}
```

where `pin` is the pin number on the Arduino board that we wish to set to a mode and `mode` can be either INPUT or OUTPUT.

digitalWrite() - Writes a HIGH or LOW value to a digital pin.

analogRead() - Reads from the analog input pin i.e., voltage applied across the pin.

Example Program for Blinking LED

Requirement

- Arduino controller board, USB connector, bread board, LED, 1.4 Kohm resistor, connecting wires, Arduino IDE.

Connection

- Connect the LED to the Arduino using the bread board and the connecting wires.

- Connect the Arduino board to the PC using the USB connector.
- Select the board type and port.
- Write the sketch in the editor, verify and upload.
- Connect the positive terminal of the LED to digital pin 12 and the negative terminal to the ground pin (GND) of Arduino board.

Sketch

```
void setup(){
pinMode(12,OUTPUT); //Set the pin mode
}
void loop(){
digitalWrite(12, HIGH); // Turn on the LED
delay(1000);
digitalWrite(12, LOW); //Turn of the LED
delay(1000);
}
```

```
void loop() {
  digitalWrite(13, state); //pin 13 equal the state value
}
void blink() {
  //ISR function
  state = !state; //toggle the state when the interrupt occurs
}
```

6.8 Case Studies

In this section, we will discuss some practical applications where Arduino board is used. Three case studies are explained – traffic control system, Digital Humidity and Temperature sensor used with Arduino and servo motor interfaced with Arduino.

6.8.1 Traffic Control System

Requirement

Arduino board, 3 different color LEDs, 330 ohm resistors and jumper wires.

Connection

- Connect the positive terminals of the LEDs to the respective digital Output pins in the board assigned in the code.
- Connect the negative terminals of the LED to the ground.

Sketch

```
// LED pins
int r=2;
int g=3;
int y=4;
void setup()
{ Serial.begin(9600);
  pinMode(r,OUTPUT);
  pinMode(g,OUTPUT);
  pinMode(y,OUTPUT);
  digitalWrite(r,LOW);
  digitalWrite(g,LOW);
  digitalWrite(y,LOW);
}
void traffic()
{
digitalWrite(g,HIGH);
Serial.println("Green LED:ON,GO");
```

```
//delay of 5 seconds
delay(5000);
digitalWrite(g,LOW);
digitalWrite(y,HIGH);
Serial.println("Green LED:OFF,Yellow LED:ON,WAIT)
delay(5000);
digitalWrite(y,LOW);
digitalWrite(r,HIGH);
Serial.println("Yellow LED:OFF, Red LED ON,STOP");
delay(5000);
digitalWrite(r,LOW);
Serial.println("ALL OFF");
}
void loop()
{
traffic();
delay(10000);
}
```

8 IoT IMPLEMENTATION WITH RASPBERRY Pi

8.1 Introduction

Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV and uses a standard keyboard and mouse. It is a little device that enables people of all ages to explore computing. Programs are written in languages like Scratch and Python. It's capable of doing everything we expect from a desktop computer. We can browse the Internet, play high-definition video, to make spreadsheets, word-processing and playing games. There are several generations of Raspberry Pi like Raspberry Pi 3 model B, Raspberry Pi 2 model B, Raspberry Pi zero.

8.2 Architecture

Figure 8.1 shows the basic architecture of Raspberry Pi. It is a low cost single-board computer which is easy to access. Table 8.1 gives a comparison of the specifications of most commonly used versions of Raspberry Pi.

Table 8.1: Comparison of the Specifications of Commonly Used Versions of Raspberry Pi

Key Features	Raspberry Pi 3 Model B	Raspberry Pi 2 Model B	Raspberry Pi Zero
RAM	1 GB SDRAM	1 GB SDRAM	512 MB SDRAM
CPU	Quad Cortex A53 @1.2 GHz	Quad Cortex A53 @900 MHz	ARM11 @ 1 GHz
GPU	400 MHz Video Core IV	250 MHz Video Core IV	250 MHz Video Core IV
Ethernet	10/100	10/100	None
Wireless	802.11/Bluetooth 4.0	None	None
Video Output	HDMI/Composite	HDMI/Composite	HDMI/Composite
GPIO	40	40	40

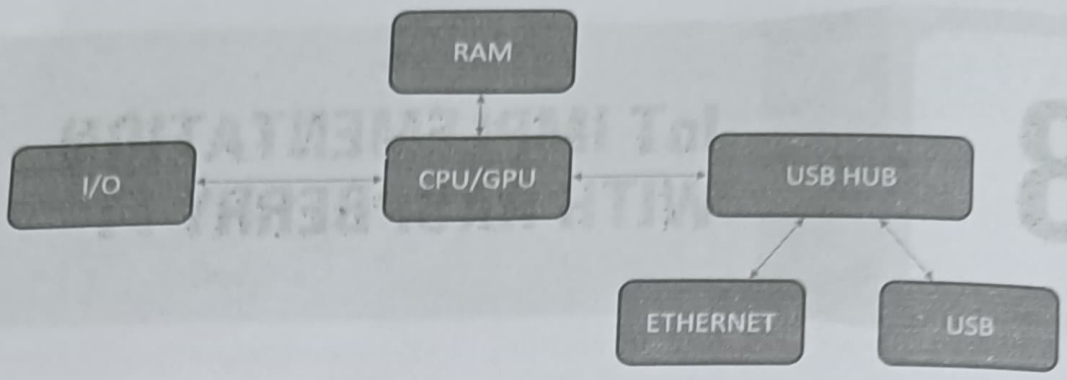

Fig.8.1: Basic Architecture of Raspberry Pi

The basic set up for Raspberry Pi includes HDMI cable, monitor, keyboard, mouse, 5 volt power adapter for Raspberry Pi, LAN cable, 2 GB micro SD card (minimum). The official operating systems supported are Raspbian and NOOBS. Other third party operating systems like Ubuntu mate, Snappy Ubuntu Core, Windows 10 Core, Pinet and Risc OS are also supported by Raspberry Pi.

For downloading Raspbian, following steps are required.

1. Download latest Raspbian image from Raspberry Pi official Website https://www.raspberrypi.org/downloads
2. Unzip the file and end up with a .img file.
3. Next step is to write Raspbian OS in SD card. For that install "Win32 Disk Imager" software in Windows machine.
4. Run Win32 Disk Imager.
5. Plug SD Card into our PC.
6. Select the "Device".
7. Browse the "Image File" (Raspbian Image).

Secure Shell (SSH) is a feature of Linux that allows you to effectively open a terminal session on your Raspberry Pi from the command line of your host computer. Recent versions of Rasbpian do not enable SSH access by default. For enabling SSH, following steps are required.

1. Open command prompt and type `sudo raspi-config` and press enter key.
2. Navigate to SSH in advanced option.
3. Enable SSH.

For expanding the file system, perform the following operations.

1. Open command prompt and type `sudo raspi-config` and press enter.
2. Navigate to Expand File System.
3. Press enter key to expand it.

Most commonly used programming languages in Raspberry Pi are Python, C, C++, Java, Scratch and Ruby. Any language that will compile for ARMV6 can be used with Raspberry Pi. The popular applications developed using Raspberry Pi are media streamer, home automation, controlling robot, Virtual Private Network (VPN), light weight Web server with IoT etc.

8.3 PIN Configuration

GPIO pins in Raspberry Pi are the general purpose Input-Output pins. These pins are to communicate with other circuitry such as extension boards, custom circuits and much more. For getting an output, we can turn a GPIO pin HIGH or LOW. Similarly for an input, it can detect a GPIO pin as HIGH or LOW. Figure 8.2 shows the pin configuration of Raspberry Pi.

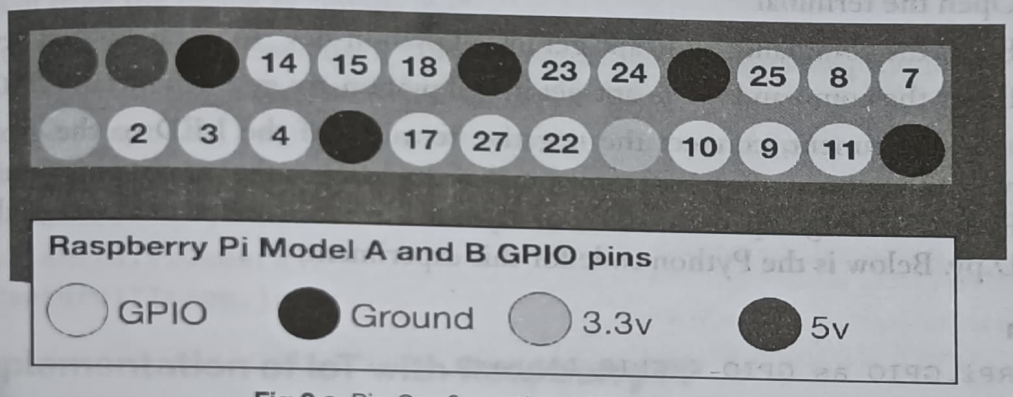

Fig.8.2: Pin Configuration of Raspberry Pi

These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Seventeen of the 26 pins are GPIO pins. Others are power or ground pins. Each pin can turn on or off, or go HIGH or LOW in computing terms. When the pin is HIGH it outputs 3.3 volts (3v3) and when the pin is LOW, it is off.

We can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch. It could be input from a sensor or a signal from another computer or device. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, we can control devices that are attached to it from anywhere and those devices can send data back. Connectivity and control of physical devices over the Internet is a powerful and exciting thing and the Raspberry Pi is ideal for this.

8.4 Case Studies

In this section, we will discuss about 2 example projects using Raspberry Pi. The first one is blinking an LED and the second one is taking a picture using PiCam. The codes for both the examples are written in Python.

8.4.1 Blinking LED

Following are the requirements for this experiment.

- Raspberry Pi
- LED
- 100 ohm resistor
- Bread board
- Jumper cables

We need to install the GPIO library. Following are the steps for installing the GPIO library.

1. Open the terminal
2. Enter the command "sudo apt-get install python-dev" to install python development.
3. Enter the command "sudo apt-get install python-rpi.gpio" to install GPIO library.

For this experiment, connect the negative terminal of the LED to the ground pin of Raspberry Pi and connect the positive terminal of the LED to the output pin of the Raspberry Pi. For Python coding, open the IDE of Python. We can write the code and save it as BlinkLED.py. Below is the Python code for this experiment.

Program

```
import RPi.GPIO as GPIO # GPIO Library
import time
# Set the board for pin numbering
GPIO.setmode(GPIO.BOARD)
# Set GPIO pin as output pin
GPIO.setup(11,GPIO.OUT)
for i in range(0,5):
    GPIO.output(11,True)#Turn on GPIO pin 11
    time.sleep(1)
    GPIO.output(11,False)
    time.sleep(2)
    GPIO.output(11,True)
GPIO.cleanup()
```

The LED blinks in a loop with delay of 1 and 2 seconds for 5 times.

8.4.2 Taking Pictures using PiCam

For this we need Raspberry Pi and a Raspberry Pi camera. There is a specific camera module which needs to be imported while writing the Python code. There is a dedicated CSI slot in Raspberry Pi for connecting the camera. The cable slot is placed between Ethernet port and

HDMI port. Boot the Raspberry Pi once the camera is connected to Raspberry Pi. Following are the steps needed to configure Raspberry Pi for camera.

- In the terminal, run the command "sudo raspi-config" and press enter key.
- Navigate to "Interfacing Options" and press enter key.
- Navigate to "camera" option.
- Enable the camera.
- Reboot Raspberry Pi.

Next step is to capture image. For this, open the terminal and enter the following command.

```
raspistill-o image.jpg
```

This will store the image as "image.jpg".

PiCam can also be processed using Python camera module python-picamera. For that we need to install the module using the following command.

```
sudo apt-get install python-picamera
```

Below is the python code for capturing the image.

```
import picamera
camera=picamera.PiCamera()
camera.capture("Image.jpg")
```

8.5 Implementation of IoT with Raspberry Pi

In this section, we will see the implementation of Raspberry Pi with Internet of Things. For this we need to integrate sensors and actuators interfaced with Raspberry Pi. The data will be read from the sensor. The actuator will be controlled according to the reading from the sensor. We will see an example of a Temperature Dependent Auto Cooling System.

8.5.1 Temperature Dependent Auto Cooling System

In this experiment a DHT sensor senses the temperature and when the temperature goes above 30° C, a fan needs to be automatically turned on.

Requirements

- DHT sensor
- 4.7 K ohm resistor
- Relay