25-01-2025 WEEK-1

AIM: Install necessary software that is necessary for Arduino.

PROCEDURE:

step 1: Go to Google Chrome and type Arduino website or Arduino cc.

step 2: Click on the hardware and explore the different types of Arduino boards.

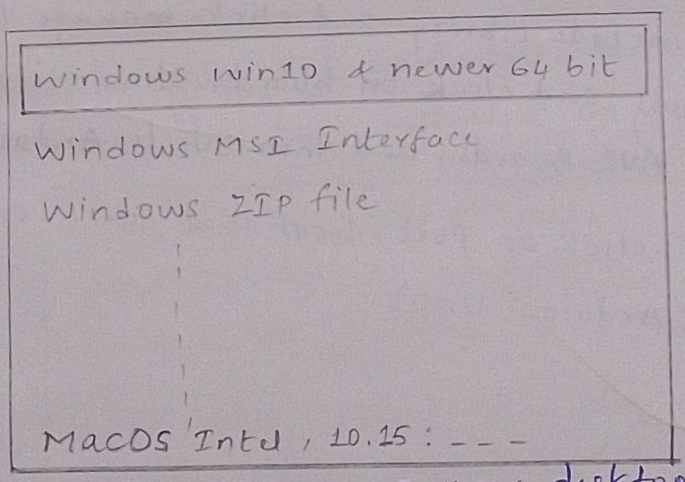
TABLE OF CONTENTS:

- Kits
- Nano Family
- MKR Family
  - Boards
  - Shields
- Classic Family
  - Boards
  - Shields
- Mega Family

step 3: Click on the software

step 4: Click on latest version of Arduino IDE and download it.

step 5: Go to download option, you will see:



step 6: Install the version on your desktop

step 7: Go to search and search the arduino IDE 2.3.4

step 8: click to open Arduino IDE 2.3.4

step 9: Go to file, click on "new sketch".

step 10: Every new file starts with month, date, followed by an alphabet.

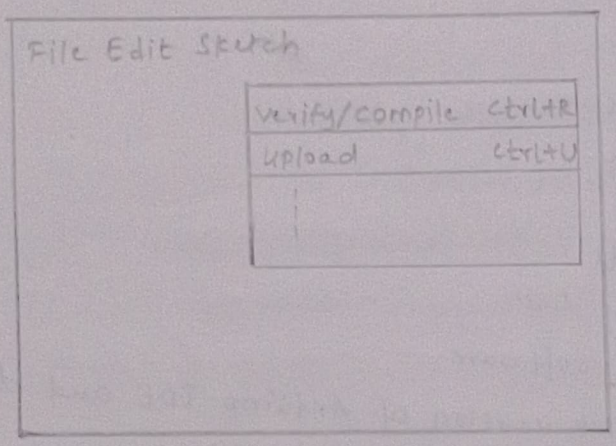
ex: sketch_jan25a → alphabet

month date

Step 11: Every file has 2 functions by default

```
void setup() {
---     // run code once
---
}
void loop() {
---
---     // main code to run repeatedly
---
}
```

Step 12: Go to sketch to run/compile a sketch



Step 13: Any external libraries to be installed, go to sketch and click on include library and click manage libraries.

Step 14: Go to tools and click on board, then click on Arduino AVR Boards, and then select Arduino UNO.

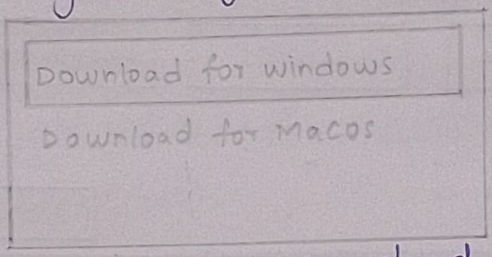
Step 15: In tools, click on port, from that select COM1 or COM3 (Arduino Uno)

* Install Software for Raspberry PI

Step 1 : Go to Google

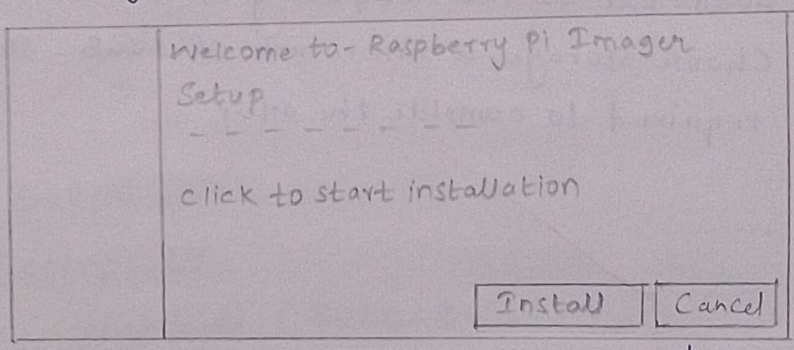
Step 2 : After the page opens, click on software.

Step 3 : When clicking on software a new page gets opened, there you can see install Raspberry PI or using Raspberry PI images.

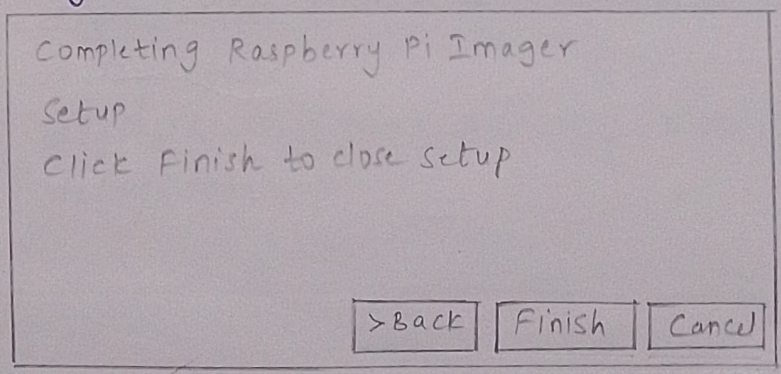


Step 4 : When we click on Download for windows, you can see in downloads, imager_1.8.5.exe "file".

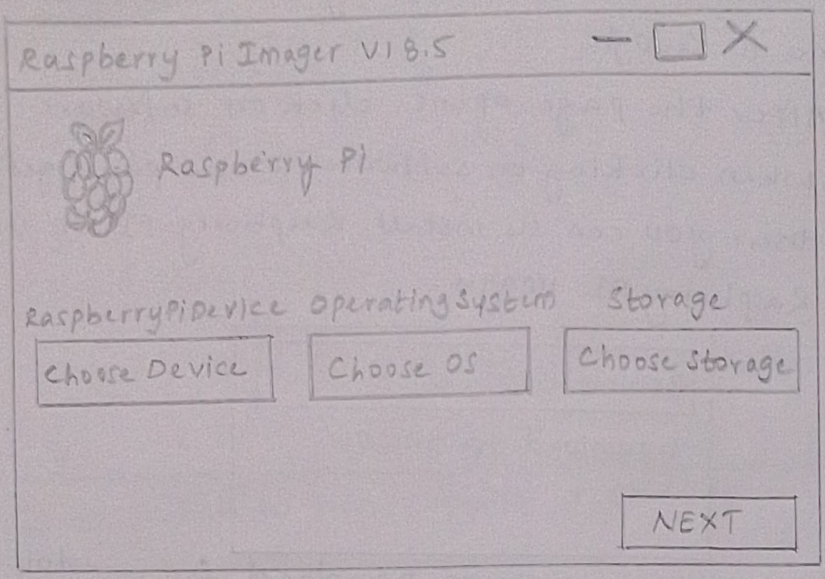
Step 5 : When you open the file imager it displays a message.



Step 6 : When you click on install, the raspberry Pi imager has been installed on your computer.



Step 7 : After the Raspberry Pi setup is completed a new page is opened.



Step 8: Choose the Raspberry Pi device (ex: Raspberry Pi 5)

Step 9: Choose the OS like Raspberry Pi OS (64-bit)

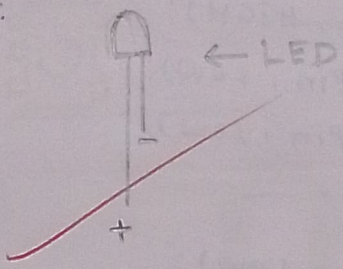
Step 10: Choose storage a minimum of 10GB. SD card is required to complete the setup.

Blink code:

Code:

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
```

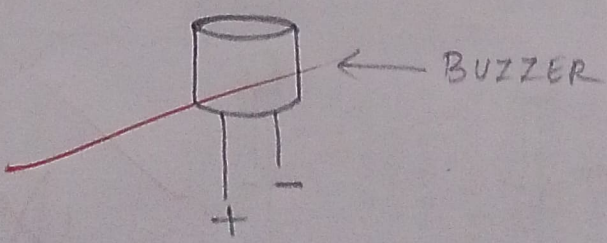
Output:



Code: Buzzer Code

```
void setup() {
    pinMode(BUZZ_BUILTIN, OUTPUT);
}
void loop() {
    digitalWrite(BUZZ_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(BUZZ_BUILTIN, LOW);
    delay(1000);
}
```

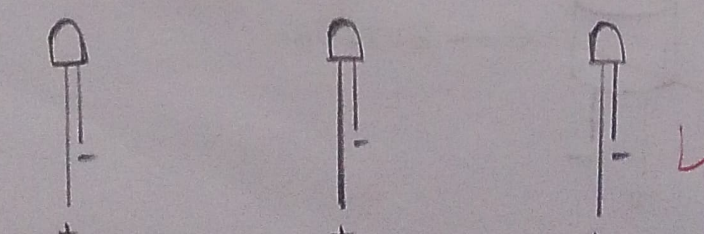
Output:



Code: For RGB

```
int redPin = 9;
int greenPin = 10;
int bluePin = 11;
void setup() {
        pinMode (redPin, OUTPUT);
        pinMode (greenPin, OUTPUT);
        pinMode (bluePin, OUTPUT);
}
void loop() {
        digitalWrite (redPin, HIGH);
        digitalWrite (greenPin, LOW);
        digitalWrite (bluePin, LOW);
        delay (1000);
        digitalWrite (redPin, LOW);
        digitalWrite (greenPin, HIGH);
        digitalWrite (bluePin, LOW);
        delay (1000);
        digitalWrite (redPin, LOW);
        digitalWrite (greenPin, LOW);
        digitalWrite (bluePin, HIGH);
        delay (1000);
        digitalWrite (redPin,  );
        digitalWrite (greenPin,  );
        digitalWrite (bluePin,  );
        delay (1000);
}
```

Output:

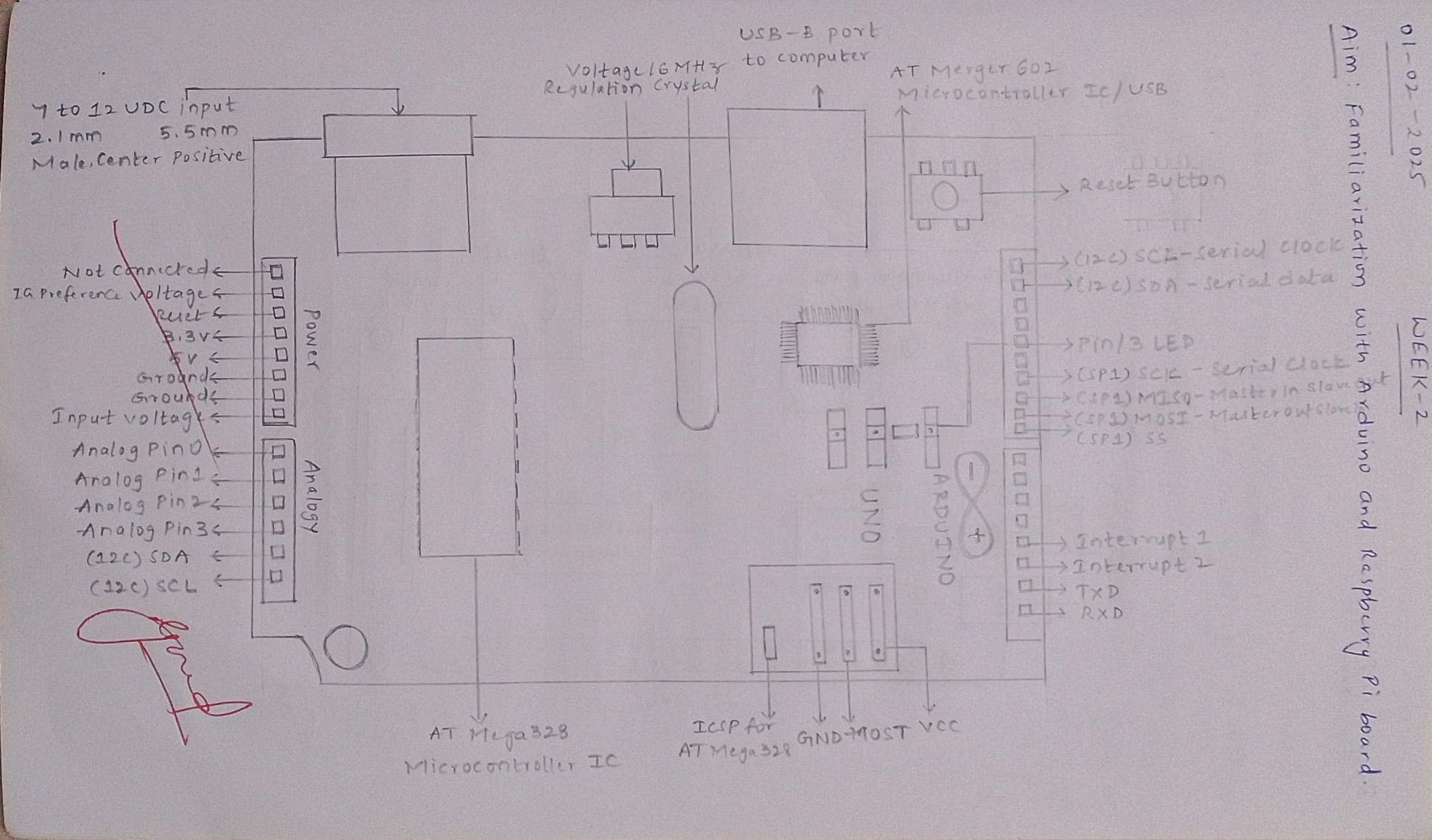


1/2/25

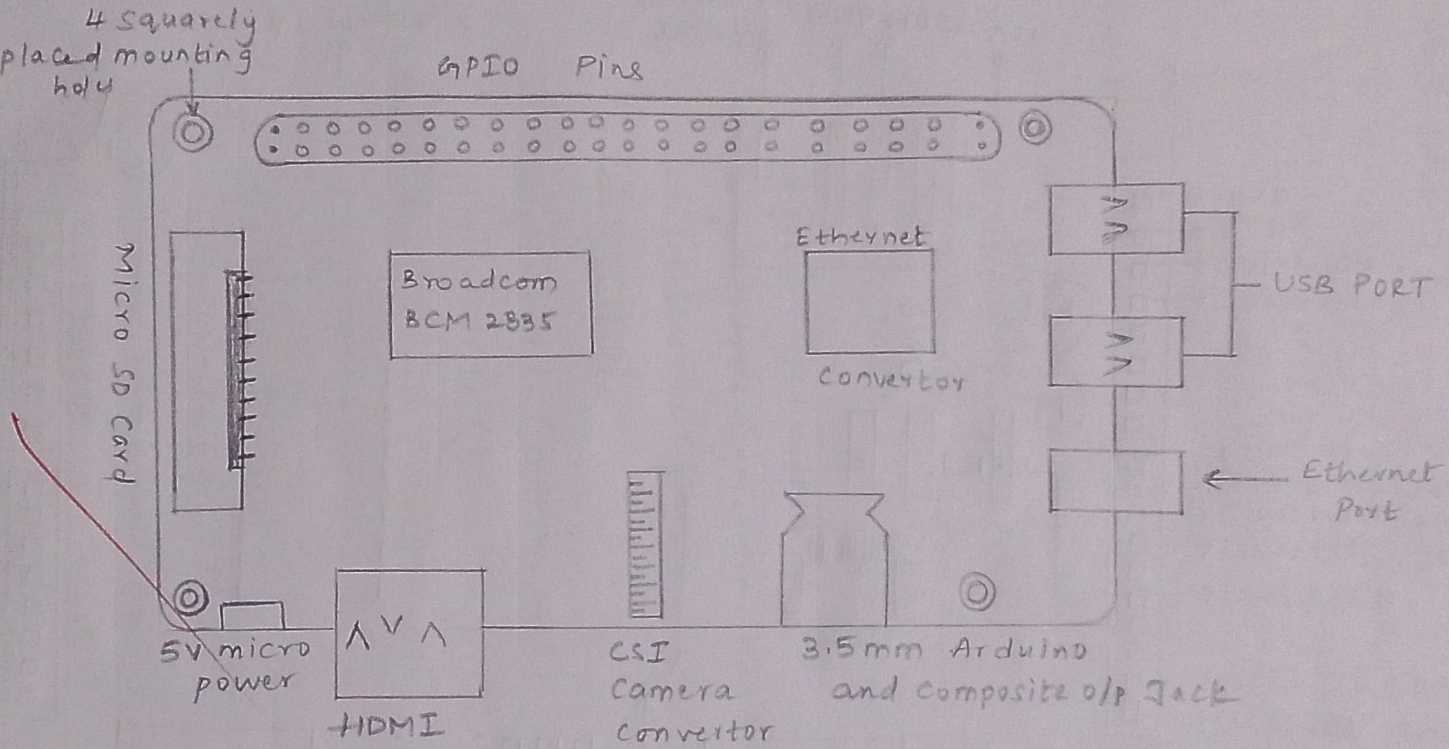
WEEK-2

01-02-2025

Aim: Familiarization with Arduino and Raspberry Pi board.



Raspberry Pi :

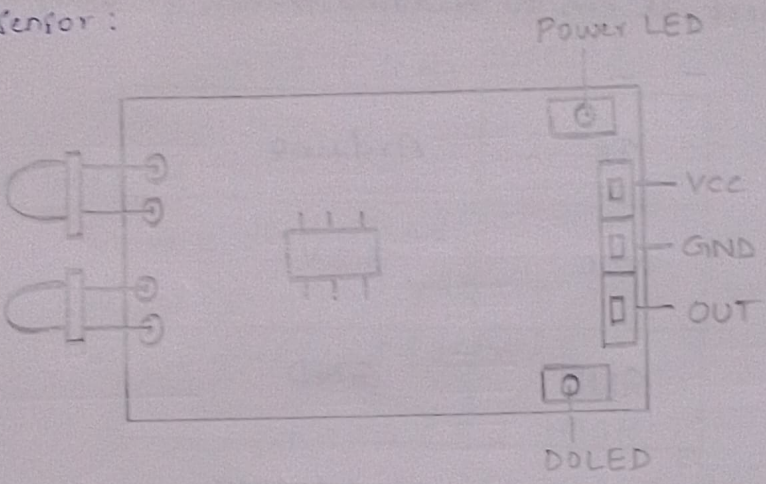


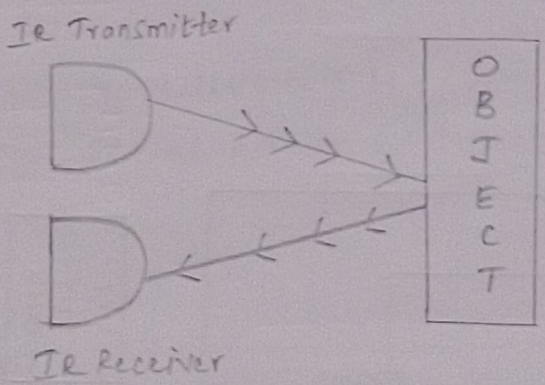
1) Write a program to interface IR sensor using Arduino board.

IR sensor → Infra red sensor / LDR sensor (Light Dependent Resistor sensor).

- It is based on the principle of optics.
- IR Radiation can be found with the wavelengths of these waves range from 0.7 to 5.

IR sensor:





Hardware Requirements:

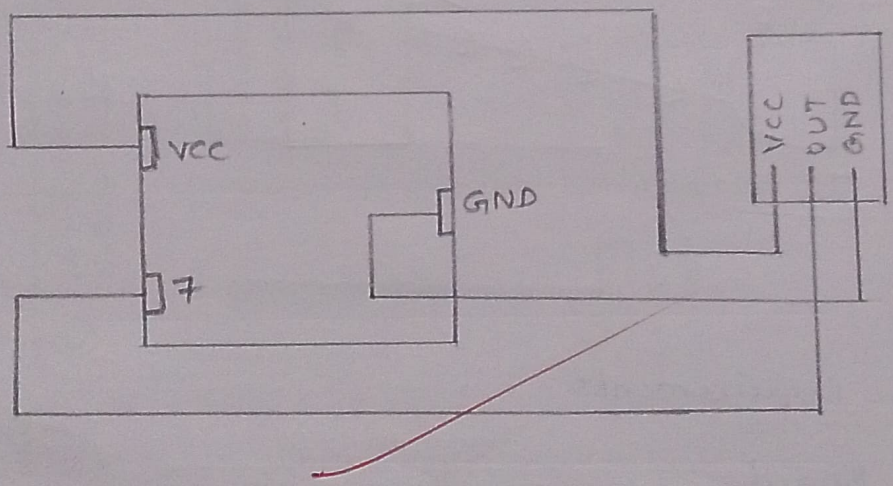
- Arduino Board
- IR sensor
- Jumper wires
- Arduino Cable

Procedure:

- Connect Arduino cable to the USB board to one end and another end to arduino board.
- Go to tools and check for the port and connect to selected arduino port.
- Select the UNO board in the board manager
- Now connect IR Pin to Arduino board to Jumper wires.

IR	Arduino
VCC	3.3 V
GND	GND
OUT	Digital IPM 0-7

Pin Diagrams:



Source Code:

```
int IRPin = 7;
int led = 13;
int value;
void setup() {
        pinMode (IRPin, INPUT);
        Serial.begin (9600);
        pinMode (led, OUTPUT);
}
void loop() {
        Value = digitalRead(IRPin);
        Serial.println (Value);
        if (digitalRead (IRPin) == 0)
        {
            digitalWrite (led, HIGH);
            Serial.println ("Object detected");
        }
        else {
            digitalWrite (led, LOW);
            Serial.println ("Object not detected");
    }
}
```

Output:

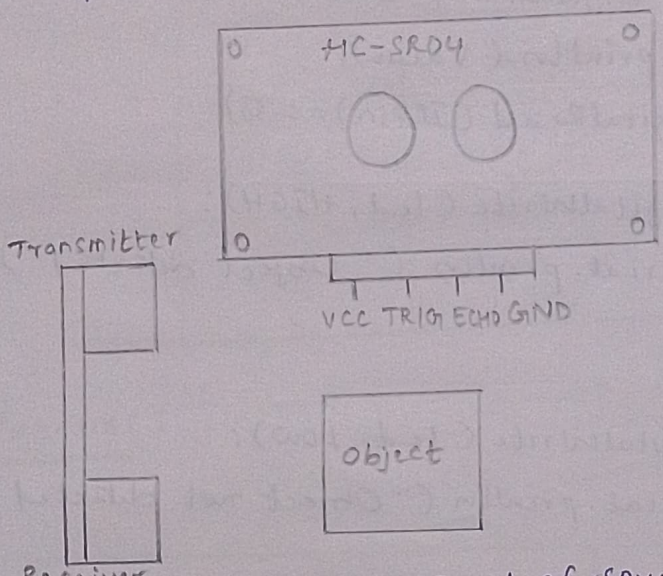
Serial Monitor

```
Object not detected
0
Object not detected
0
Object not detected
0
Object detected
1
Object detected
1
Object detected
1
```

2) Write a program to interface ultrasonic sensor using Arduino board.

Ultrasonic sensor:

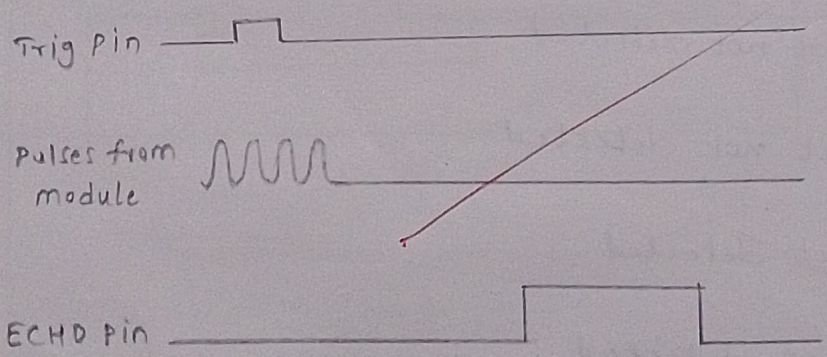
- It is an instrument that measures the distance to an object using ultrasonic sound waves.
- It uses a transducer to send and receive ultrasonic pulses that relay back the information about an object.
- It converts a sound signal into electrical signal
- Principle of ultrasonic



Distance = Time x speed of sound / 2

Time can be calculated by

Ultrasonic HC-SR04 module Timing Diagram

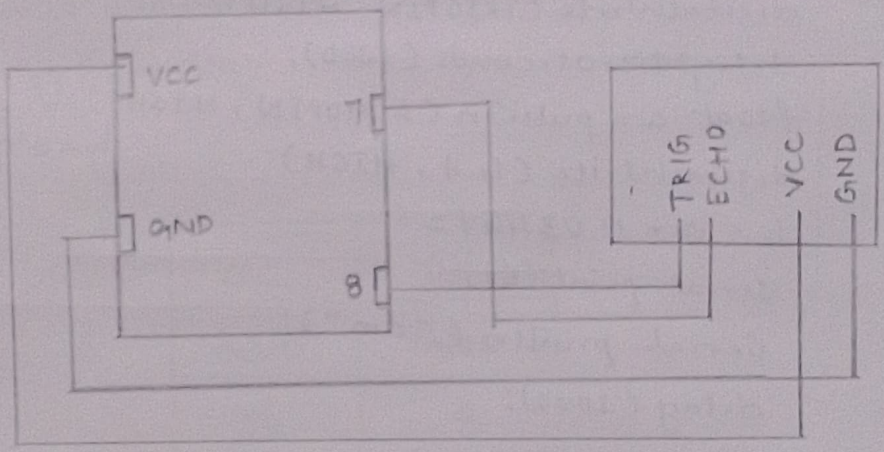


Hardware Requirements

- Arduino UNO
- Arduino Cable
- Ultrasonic Sensor HC-SR04
- Jumper wires
- LED

VCC	5V/3.3V
GND	GND
TRIG	7
ECHO	8

Pin Diagram :



Procedure :

- Connect Arduino cable to the USB board to one end and another end to the arduino board.
- Go to tools, select port and select communication board.
- Connect jumper wires to the pins.
- Verify and upload.
- Check the output in the serial monitor

Source Code:

```
#define ECHOPIN 7
#define TRIGPIN 8
int led = 12;
int a, b;
void setup() {
        Serial.begin(9600);
        pinMode(ECHOPIN, INPUT);
        pinMode(TRIGPIN, OUTPUT);
        pinMode(led, OUTPUT);
}
void loop() {
        digitalWrite(TRIGPIN, LOW);
        delayMicroseconds(2000);
        digitalWrite(TRIGPIN, HIGH);
        delayMicroseconds(1000);
        float a = pulseIn(ECHOPIN, HIGH);
        digitalWrite(led, HIGH);
        b = a * 0.0344/2;
        Serial.print(6);
        Serial.println("cm");
        delay(1000);
}
```

Output:

Serial Monitor

0 cm
2067 cm
27 cm
14 cm
8 cm
5 cm
11 cm
110 cm
2323 cm

8/2/25

08-02-2025 WEEK-3

Aim: Write a program to transfer sensor data to smaul-phone using Bluetooth on Arduino.

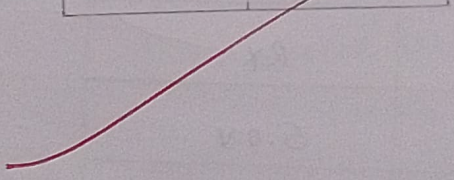
Hardware Requirements:

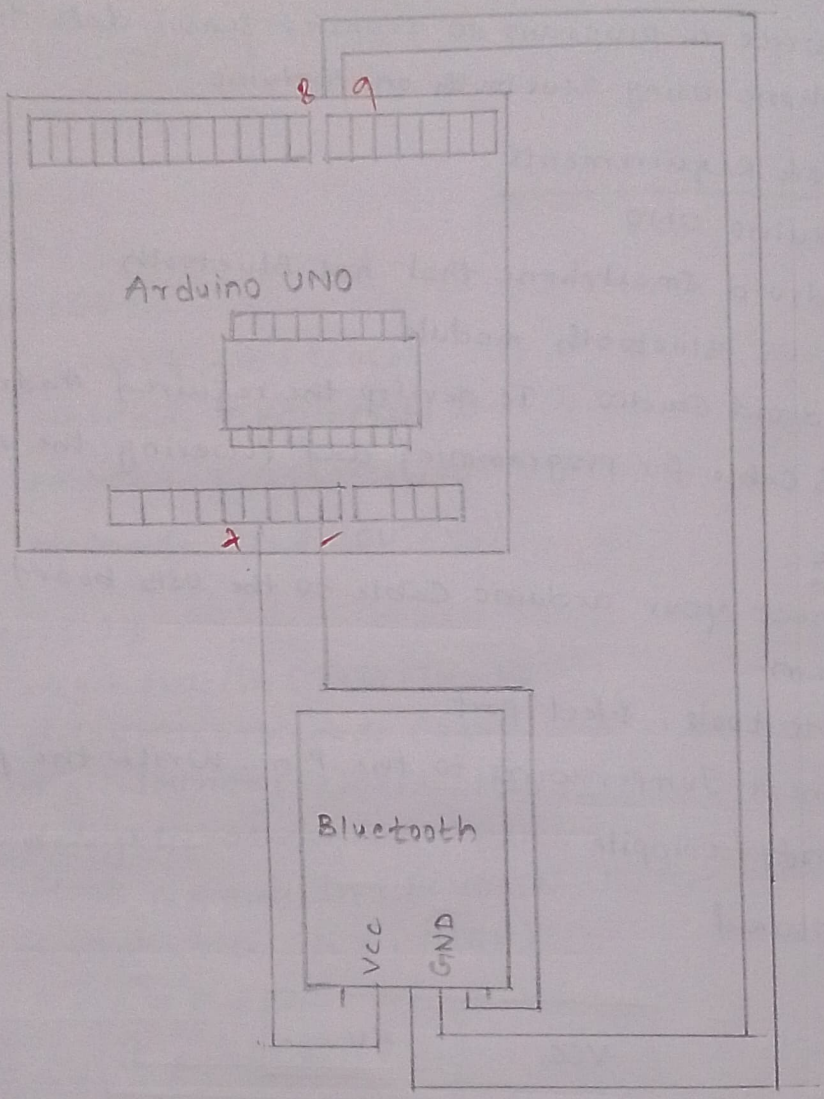
- Arduino UNO
- Android Smaulphone that has Bluetooth.
- Hc-05 Bluetooth module
- Android Studio (To develop the required Android app)
- USB Cable for programming and Powering the Arduino

procedure:

- Connect your arduino Cable to the USB board of the system.
- Go to tools, select port.
- Connect Jumper wires to the Pin. Write the program.
- verify / compile.
- Upload.

VCC	5V
GND	GND
Tx	8
Rx	9





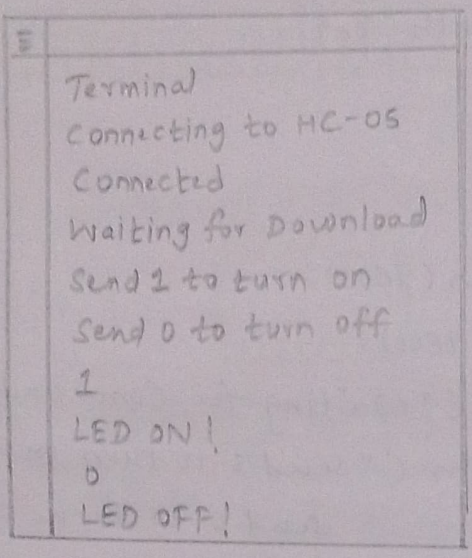
Arduino UNO	BT Sensor
9	TX
8	RX
VCC	5.0 V
GND	GND

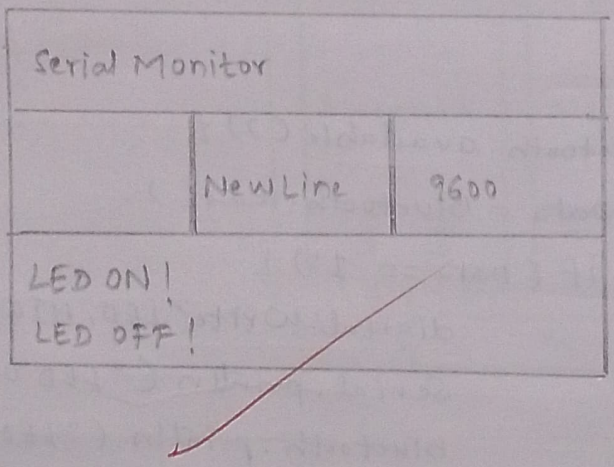
Source Code:

```
#include <SoftwareSerial.h>
SoftwareSerial Bluetooth(8, 9);
int LED = 13;
int Data;
void setup() {
    Bluetooth.begin(9600);
    Serial.begin(9600);
    Serial.println("Waiting for Command...");
    Bluetooth.println("Send 1 to turn on the LED
                       Send 0 to turn off");

    pinMode(LED, OUTPUT);
}
void loop() {
    if(Bluetooth.available()) {
        Data = Bluetooth.read();
        if(Data == '1') {
            digitalWrite(LED, HIGH);
            Serial.println("LED ON!");
            Bluetooth.println("LED ON!");
        }
        else if(Data == '0') {
            digitalWrite(LED, LOW);
            Serial.println("LED OFF!");
            Bluetooth.println("LED OFF!");
        }
        else { ; }
    }
    delay(1000);
}
```

Output:





15/9/25

WEEK-4

Aim: Write a program to implement RFID using Arduino

SPI Protocol:

- SPI is a Serial Peripheral Interface
- It is a serial communication protocol often used in embedded systems for high speed data exchange between devices on the bus.
- It operates using master-slave.

  signals:

  1. a clock (SSLK)
  2. a master o/p / slave i/p (MOSI)
  3. a master i/p / slave o/p (MISO)
  4. a slave select (SS)

- The master pulls low on a slave line.
- It supports full duplex communication
- when master and slave can transmit the data simultaneously
- The data speed reaches to 100MHz.
- The first line shows the firm wave version of the MFRG522TC.
- In this case the result is 0x92.
- A typical 1K RFID tag has 1K byte of memory organised into 16 sectors, each sector consists of 4 blocks.

Sector	Block	Byte Number within a Block														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A						Access Bits				Key B					
	2																
	1																
	0																
14	3	Key A						Access Bits				Key B					
	2																
	1																

- Block '0' of sector '0' is reserved for storing manufacturer data.
- It contains 4 byte unique ID.
- Each sector consists of 3 data blocks.
- Which can be used for storing user data and is known as sector trailer.
- 16 sector contains 16 sector trailers.
- And each sector trailer consists of key. A 6 bytes mandatory.

  KeyB optional – 6 bytes

  4 bytes for access bits

Hardware Requirements:

- Arduino
- Jumper wires
- USB Cable
- RC522 RFID reader
- Cord
- Key chain

RST	9
SDA(SS)	10
MOSI	11
MISO	12
SCK	13

Source Code:

demoinfo.ino:

```
#include<SPI.h>
#include<MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
        Serial.begin(9600);
        while (!Serial);
        SPI.begin();
        mfrc522.PCD_Init();
        delay(4);
        mfrc522.PCD_DumpVersionToSerial();
        Serial.println(F("Scan PICC to see UID, SAK,
                          type and data blocks..."));
}
```

```
void loop() {
    if (!mfrc522.PICC_IsNewCardPresent()) {
        return;
    }
    if (!mfrc522.PICC_ReadCardSerial()) {
        return;
    }
    mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

readinfo.ino :

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SSPIN, RST_PIN);
void setup() {
    Serial.begin(9600);
    SPI.begin();
    mfrc522.PCD_Init();
    Serial.Println(F("Read Personal data on a
                    MIFARE PICC:"));
}
void loop() {
    MFRC522::MIFARE_Key;
    for(byte i=0; i<6; i++) key.keyByte[i] = 0xFF;
    byte block;
    byte len;
    MFRC522::StatusCode status;
    if (!mfrc522.PICC_IsNewCardPresent()) {
        return;
    }
    if (!mfrc522.PICC_ReadCardSerial()) {
        return;
    }
    Serial.Println(F("** Card Detected: **"));
    mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));
    Serial.Print(F("Name: "));
```

```
byte buffer1[18];
block = 4;
len = 18;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_
        AUTH_Key_A, 4, &Key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
        Serial.Print(F("Authentication Failed: "));
        Serial.Println(mfrc522.GetStatusCodeName(status));
        return;
}
status = mfrc522.MIFARE_Read(block, buffer1, &len);
if(status != MFRC522::STATUS_OK) {
        Serial.Print(F("Reading Failed: "));
        Serial.Println(mfrc522.GetStatusCodeName(status));
        return;
}
for(byte i=0; i<16; i++) {
        Serial.Write(buffer2[i]);
}
Serial.Println(F("/n **End Reading **/n"));
delay(1000);
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto();
}
```

writeinfo.ino:

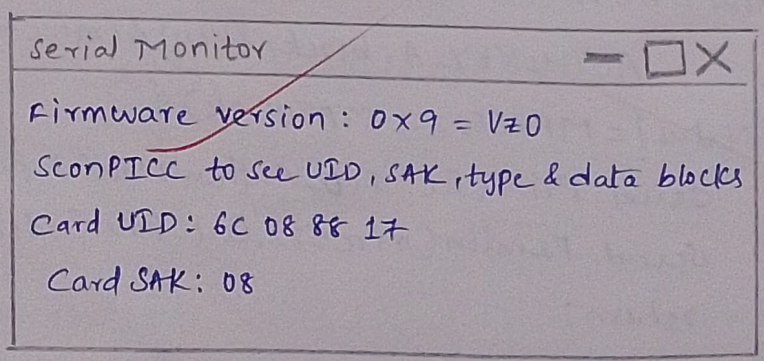
```
#include<SPI.h>
#include<MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
    Serial.begin(9600);
    SPI.begin();
    mfrc522.PCD_Init();
    Serial.Println(F("Write Personal data on a MIFARE
                    PICC"));
}
```

```
void loop() {
    MFRC522::MIFARE_Key key;
    for (byte i=0; i<6; i++)
        key.keyByte[i] = 0xFF;
    if (!mfrc522.PICC_IsNewCardPresent()) {
        return;
    }
    if (!mfrc522.PICC_ReadCardSerial()) {
        return;
    }
    Serial.print(F("Card UID: "));
    for (byte i=0; i<mfrc522.uid.size; i++) {
        Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
        Serial.print(mfrc522.uid.uidByte[i], HEX);
    }
    Serial.print(F("PICC type: "));
    MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
    Serial.println(mfrc522.PICC_GetTypeName(piccType));
    byte buffer[34];
    byte block;
    MFRC522::StatusCode status;
    byte len;
    Serial.setTimeout(20000L);
    Serial.println(F("Type First Name ending with #"));
    len = Serial.readBytesUntil('#', (char*)buffer, 20);
    for (byte i=len; i<20; i++) buffer[i] = ' ';
    block = 1;
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF
        AUTH_KEY_A, block, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("PCD_Authenticate() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }
    status = mfrc522.MIFARE_Write(block, buffer, 16);
```

```
if (status != MFRC522::STATUS_OK) {
    Serial.Print(F("MIFARE_write() failed: "));
    Serial.Println(mfrc522.GetStatusCodeName(status));
    return;
}
else Serial.Println(F("MIFARE_write() success: "));
block = 2;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF
        _AUTH_KEY_A, block, &key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
    Serial.Print(F("PCD_Authenticate() failed: "));
    Serial.Println(mfrc522.GetStatusCodeName(status));
    return;
}
status = mfrc522.MIFARE_Write(block, &buffer[16], 10);
if (status != MFRC522::STATUS_OK) {
    Serial.Print(F("MIFARE_write() failed: "));
    Serial.Println(mfrc522.GetStatusCodeName(status));
    return;
}
else Serial.Println(F("MIFARE_write() success: "));
Serial.Println(" ");
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto();
}
```

Output:

demoinfo.ino:



PICC_type : MIFARE1KB

Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	Access Bits
15	63	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	[0 0 1]
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	61	08	88	17	68	89	62	64	65	66	67	68	69	[0 0 0]

Read → output:

After reading the card

```
** Card Detected **
Card UID : 5C 22 DA CC
Card SAK : 08
PICC type : MIFARE 1KB
Authentication failed
Reading Failed
```

Writing data on a MIFARE PICC then reading the data

```
** Card Detected **
Card UID : 5C 22 DA CC
Card SAK : 08
PICC type : MIFARE 1KB
Name : Ashvith
*End Reading*
```

Write:

```
Write Personal data on a MIFARE PICC
Card UID : 5C LL DA CC PICC type : MIFARE 1KB
Type FirstName, ending with #
PCD_Authenticate() success :
MIFARE_Write() success :
MIFARE_Write() success (Arduino #)
Type first name ending with #
MIFARE_Write() Success :
MIFARE_Write() Success :
```

PICC: Proximity Integrated Circuit Cards that serve as different electromagnetic field coupling between reader and the card.

PCD: Proximity Coupling Device also known as RFID. They decode the RFID tags and communicate with them based on ISO1444 3 standard. PCD can perform read and write operation of data.

PCD ensures the generation of a magnetic field whereas the antenna of PICC allows receiving the magnetic field.

- The frequency generated by MFRC522 is 13.56 MHz.
- The PICC_HaltA() function sends a halt command to the RFID Card which stops further communication with card.
- The PCD_StopCrypto() function stops the encryption of the data between the RFID Card and the reader.

Arne
22/2/25