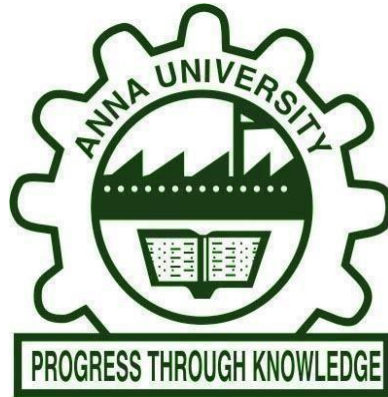


**ANNA UNIVERSITY REGIONAL CAMPUS**  
**COIMBATORE - 641 046.**



**CS3691 EMBEDDED SYSTEMS AND IOT LABORATORY**

**Name:** .....

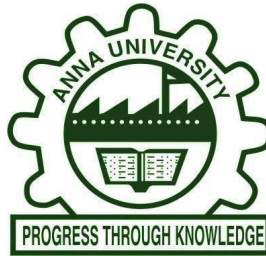
**Register No.:** .....

**Degree & Branch:** .....

**Semester:** .....

**Subject Code & Title:** .....

**ANNA UNIVERSITY REGIONAL CAMPUS  
COIMBATORE – 641 046.**



**Register No.:** \_\_\_\_\_

**BONAFIDE CERTIFICATE**

Certified to be the BONAFIDE RECORD work done by

Mr./ Ms. \_\_\_\_\_ of VI Semester,

B.Tech. Artificial Intelligence and Data science Discipline in the CS3691

Embedded Systems and IoT Laboratory Practical Course during the

Academic year 2024- 2025.

Date .....

**Staff in charge**

**Head of the Department**

Submitted for the University Practical Examination held on .....

**Internal Examiner**

**External Examiner**

## TABLE OF CONTENTS

[illegible]

[illegible]

<b>EXP NO. :</b> 01	<b>8051 ASSEMBLY LANGUAGE EXPERIMENT USING SIMULATOR</b>
<b>DATE:</b>	

**AIM:**

To write an 8051 Assembly language experiment using simulator.

**APPARATUS REQUIRED:**

S.NO.	NAME OF COMPONENT	SPECIFICATION	QUANTITY
1.	Keil $\mu$ vision 5 software	Version 5	1
2.	PC with Windows	Windows 7	1

**PROCEDURE:**

- Step 1 : Create a new project in Keil  $\mu$  Vision & Select the device AT89C51.
- Step 2 : Add the code to a new C file in the project & Save it as .asm file
- Step 3 : Debug the Code using Debug  $\rightarrow$  Start / Stop Debug Session.
- Step 4 : Stop the Program.

**PROGRAM:**

```

ORG
0000H
CLR C
MOV A, #13H
ADD A, #12H
MOV R0, A
END

```

## OUTPUT:

The screenshot displays an 8051 assembly editor with two main panels. The left panel, titled 'Registers', shows a table of registers and their values. The right panel, titled 'add.asm', shows the assembly code being executed.

Register	Value
<b>Regs</b>	
r0	0x25
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
<b>Sys</b>	
a	0x25
b	0x12
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0008
states	5
sec	0.00000250
psw	0x01

```
1  ORG 000H
2  MAIN:
3  MOV A, #13H
4  MOV B, #12H
5  ADD A, B
6  MOV R0, A
7  END
```

## RESULT:

Thus the 8051-assembly language program is written and executed successfully.

<b>EXP NO. :</b> 02	<b>TEST DATA TRANSFER BETWEEN REGISTERS AND MEMORY</b>
<b>DATE:</b>	

**AIM:**

To execute an Assembly language program to transfer data between registers and memory.

**APPARATUS REQUIRED**

S.NO.	NAME OF COMPONENT	SPECIFICATION	QUANTITY
1.	Keil $\mu$ vision 5 software	Version 5	1
2.	PC with Windows	Windows 7	1

**PROCEDURE:**

Step 1 : Create a new project in Keil  $\mu$  Vision & Select the device AT89C51.

Step 2 : Add the code to a new C file in the project & Save it as .asm file.

Step 3 : Debug the Code using Debug  $\rightarrow$  Start / Stop Debug Session.

Step 4 : Stop the Program.

**PROGRAM:**

```

ORG 0000H

CLR C

MOV R0, #10H

MOV R1, #20H

MOV R7, #08H

BACK: MOV A, @R0

MOV @R1, A

INC R0

INC R1

DJNZ R7, BACK

END

```

## OUTPUT:

The screenshot displays an assembly simulator interface. On the left, a 'Registers' window shows the state of various registers. On the right, an 'exp2.asm' window shows the assembly code being executed.

**Registers Window:**

Register	Value
<b>Regs</b>	
r0	0x18
r1	0x28
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
<b>Sys</b>	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x000D
states	52
sec	0.00002600
psw	0x00

**exp2.asm Window:**

```
1  ORG 0000H
2  CLR C
3  MOV R0, #10H
4  MOV R1, #20H
5  MOV R7, #08H
6  BACK:MOV A, @R0
7  MOV @R1, A
8  INC R0
9  INC R1
10 DJNZ R7, BACK
11 END
```

## RESULT:

Thus, Assembly language program to transfer data between registers and memory is written and executed successfully.



<b>EXP NO. :</b> 03	<b>ALU PROGRAM</b>
<b>DATE:</b>	

**AIM:**

To write and execute the ALU program using the Keil simulator.

**APPARATUS REQUIRED:**

S.NO.	NAME OF COMPONENT	SPECIFICATION	QUANTITY
1.	Keil $\mu$ vision 5 software	Version 5	1
2.	PC with Windows	Windows 7	1

**PROCEDURE:**

- Step 1 : Create a new project in Keil  $\mu$  Vision & Select the device AT89C51.
- Step 2 : Add the code to a new C file in the project & Save it as .asm file
- Step 3 : Debug the Code using Debug → Start / Stop Debug Session.
- Step 4 : Stop the Program.

**THEORY**

An arithmetic-logic unit is the part of a central processing unit that carries out arithmetic and logic operations on the operands in computer instruction words. In some processors, the ALU is divided into two units: an arithmetic unit (AU) and a logic unit (LU). Some processors contain more than one AU -- for example, one for fixed-point operations and another for floating-point operations.

Typically, the ALU has direct input and output access to the processor controller, main memory (random access memory or RAM in a personal computer) and input/output devices. Inputs and outputs flow along an electronic path that is called a bus. The input consists of an instruction word, sometimes called a machine instruction word, that contains an operation code or "opcode," one or more operands and sometimes a format code. The operation code tells the ALU what operation to perform and the operands are used in the operation.

## PROGRAM:

### ADDITION :

ORG 000H

MAIN:

MOV A, #25H

MOV B, #10H

ADD A, B

MOV R0, A

END

### OUTPUT:

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Registers', shows a list of registers and their current values. The right panel, titled 'Disassembly', shows the assembly code being executed, with the current instruction highlighted.

**Registers Panel:**

Register	Value
r0	0x35
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x35
b	0x10
sp	0x07
sp_max	0x07
dptr	0x0000
PC	0x0008
states	5
sec	0.00000250
psw	0x00

**Disassembly Panel:**

Address	Instruction
C:0x0008	00 NOP
C:0x0009	00 NOP
C:0x000A	00 NOP
C:0x000B	00 NOP
C:0x000C	00 NOP
C:0x000D	00 NOP

The assembly code being executed is shown in the 'add.asm' file:

```
1 ORG 0000H
2 MAIN:
3 MOV A, #25H
4 MOV B, #10H
5 ADD A, B
6 MOV R0, A
7 END
```

## SUBTRACTION :

ORG 000H

MAIN:

MOV A, #25H

MOV B, #20H

SUBB A, B

MOV R0, A

END

## OUTPUT:

Register	Value
Regs	
r0	0x05
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x05
b	0x20
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0008
states	5
sec	0.00000250
psw	0x00

Address	Hex	Dec	Op
C:0x0008	00		NOP
C:0x0009	00		NOP
C:0x000A	00		NOP
C:0x000B	00		NOP
C:0x000C	00		NOP
C:0x000D	00		NOP

**subb.asm**  
1 ORG 0000H  
2 MAIN:  
3 MOV A, #25H  
4 MOV B, #20H  
5 SUBB A, B  
6 MOV R0, A  
7 END

## MULTIPLICATION:

ORG 000H

MAIN:

MOV A, #2H

MOV B, #4H

MUL AB

MOV R0, A

END

## OUTPUT:

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Registers', shows a list of registers and their values. The right panel, titled 'Disassembly', shows the assembly code being executed.

**Registers Panel:**

Register	Value
Regs	
r0	0x08
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x08
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0007
states	8
sec	0.00000400
psw	0x01

**Disassembly Panel:**

Address	Disassembly
C:0x0007	00 NOP
C:0x0008	00 NOP
C:0x0009	00 NOP
C:0x000A	00 NOP
C:0x000B	00 NOP
C:0x000C	00 NOP

**mul.asm File:**

```
1 ORG 0000H
2 MAIN:
3 MOV A, #2H
4 MOV B, #4H
5 MUL AB
6 MOV R0, A
7 END
```

## DIVISION:

ORG 000H

MAIN:

MOV A, #48H

MOV B, #4H

DIV AB

MOV R0, A

END

## OUTPUT:

The screenshot displays a microcontroller simulator interface. On the left, a 'Register' window shows the state of various registers. The 'Regs' section includes r0 through r7, with r0 containing 0x12. The 'Sys' section includes a, b, sp (0x07), sp\_max, dptr, PC \$ (C:0x0007), states (8), sec (0.00000400), and psw (0x00). On the right, the assembly code for 'div.asm' is shown, with line 1 'ORG 0000H' highlighted. Below the code, a memory dump shows addresses C:0x0007 through C:0x000C, all containing the value 00.

Register	Value
r0	0x12
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x12
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0007
states	8
sec	0.00000400
psw	0x00

```
1 ORG 0000H
2 MAIN:
3 MOV A, #48H
4 MOV B, #4H
5 DIV AB
6 MOV R0, A
7 END
```

Address	Value	Operation
C:0x0007	00	NOP
C:0x0008	00	NOP
C:0x0009	00	NOP
C:0x000A	00	NOP
C:0x000B	00	NOP
C:0x000C	00	NOP

**OR :**

ORG 000H

MAIN:

MOV A, #25H

MOV B, #15H

ORL A, B

MOV 47H, A

END

## OUTPUT

The screenshot displays a microcontroller development environment with three main panels:

- Registers Panel:** A table showing the state of various registers.
- Disassembly Panel:** A list of instructions with their memory addresses and hex values.
- Assembly Panel:** The source code file named 'OR.asm'.

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x35
b	0x15
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0009
states	5
sec	0.00000250
psw	0x00

Address	Hex	Instruction
C:0x0009	00	NOP
C:0x000A	00	NOP
C:0x000B	00	NOP
C:0x000C	00	NOP
C:0x000D	00	NOP
C:0x000E	00	NOP

**OR.asm**

```
1 ORG 0000H
2 MAIN:
3 MOV A, #25H
4 MOV B, #15H
5 ORL A, B
6 MOV 47H, A
7 END
```

## XOR :

ORG 000H

MAIN:

MOV A, #45H

MOV B, #67H

XRL A, B

MOV 48H, A

END

## OUTPUT:

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Registers', shows a list of registers and their values. The right panel, titled 'Disassembly', shows the assembly code for the file 'XOR.asm'.

**Registers Panel:**

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x22
b	0x67
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0009
states	5
sec	0.00000250
psw	0x00

**Disassembly Panel:**

Address	Op-Code	Comment
C:0x0009	00	NOP
C:0x000A	00	NOP
C:0x000B	00	NOP
C:0x000C	00	NOP
C:0x000D	00	NOP
C:0x000E	00	NOP

**XOR.asm File Content:**

```
1 ORG 0000H
2 MAIN:
3 MOV A, #45H
4 MOV B, #67H
5 XRL A, B
6 MOV 48H, A
7 END
```

**RESULT:**

Thus, the ALU program using the Keil simulator is written and executed successfully.



**EXP NO. : 04**

**DATE:**

**BASIC ARITHMETIC PROGRAM USING EMBEDDED C**

**AIM:**

To write a basic arithmetic Program using Embedded C

**APPARATUS REQUIRED:**

S.NO.	NAME OF COMPONENT	SPECIFICATION	QUANTITY
1.	Keil $\mu$ vision 5 software	Version 5	1
2.	PC with Windows	Windows 7	1

**PROCEDURE:**

- Step 1 : Create a new project in Keil  $\mu$  Vision & Select the device AT89C51.
- Step 2 : Add the code to a new C file in the project & Save it as .asm file
- Step 3 : Debug the Code using Debug  $\rightarrow$  Start / Stop Debug Session.
- Step 4 : Stop the Program.

**PROGRAM:**

```
#include<REG51.H>
unsigned char a, b;
void main()
{
a=0x06;
b=0x03;
P0=a-b;
P1=a+b;
P2=a*b;
P3=a/b;
while(1);
```

## OUTPUT:

```
1 #include<REG51.H>
2 unsigned char a, b;
3 void main()
4 {
5     a=0X06;
6     b=0X03;
7     P0=a-b;
8     P1=a+b;
9     P2=a*b;
10    P3=a/b;
11    while(1);
12 }
13
```

The output shows the results of the program execution for four parallel ports. Each window displays the port number, the value of the port register (P0, P1, P2, P3), and the value of the pins (Pins).

Port	Value	Pins
Port 0	0x03	0x03
Port 1	0x09	0x09
Port 2	0x12	0x12
Port 3	0x02	0x02

## RESULT:

Thus, a basic arithmetic Program using Embedded C was written and executed successfully.

<b>EXP NO.:</b> 05	<b>INTRODUCTION TO ARDUINO PLATFORM AND PROGRAMMING</b>
<b>DATE:</b>	

**AIM:**

To study the basics of Arduino Uno board and Arduino IDE 2.0 software.

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Hardware & Software Requirement	Quantity
1	Arduino IDE 2.0	1
2	Arduino UNO Board	1
3	Jump wires	Few
4	Arduino USB Cable	1

**INTRODUCTION TO ARDUINO:**

Arduino is a project, open-source hardware, and software platform used to design and build electronic devices. It designs and manufactures microcontroller kits and single-board interfaces for building electronics projects. The Arduino boards were initially created to help students with the non-technical background. The designs of Arduino boards use a variety of controllers and microprocessors. Arduino is an easy-to-use open platform for creating electronic projects. Arduino boards play a vital role in creating different projects. It makes electronics accessible to non-engineers, hobbyists, etc. The various components present on the Arduino boards are a Microcontroller, Digital Input/output pins, USB Interface and Connector, Analogue Pins, reset buttons, Power buttons, LEDs, Crystal oscillators, and Voltage regulators. The most standard and popular board used over time is Arduino UNO. The ATmega328 Microcontroller present on the UNO board makes it rather powerful than other boards. There are various types of Arduino boards used for different purposes and projects. The Arduino Boards are organized using the Arduino (IDE), which can run on various platforms. Here, IDE stands for Integrated Development Environment.

## ARDUINO DUE:

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 processor, making it the first Arduino to use a 32-bit ARM core. Operating at 3.3V with an 84 MHz clock speed, it offers higher processing power compared to 8-bit boards. It features 54 digital I/O pins, 12 analog inputs, 2 analog outputs (DAC), and USB host capabilities. With 512 KB of flash memory and 96 KB of SRAM, it is ideal for advanced projects involving data processing, robotics, and signal generation. However, care must be taken as its pins are not 5V tolerant, unlike most Arduino boards.



## PROGRAM:

```
void setup()
{
  pinMode(4,OUTPUT);
}

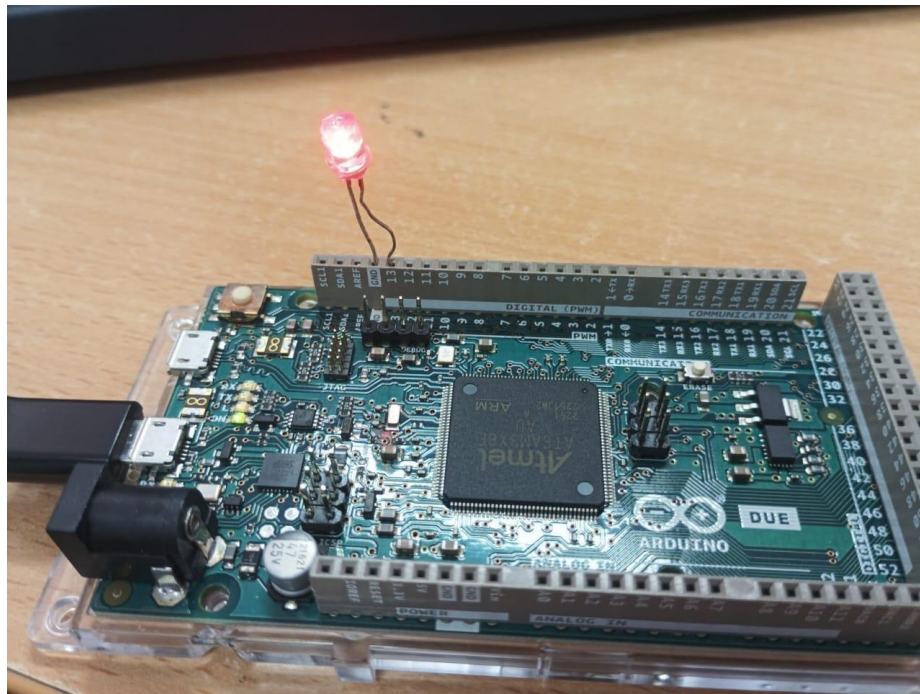
void loop() {
  digitalWrite(4,HIGH);

  delay(1000);

  digitalWrite(4, LOW);

  delay(1000);
}
```

## OUTPUT:



**RESULT:**

Thus the study of Arduino DEU board and Arduino IDE platform is done successfully.

<b>EXP NO.:</b> 06	<b>EXPLORE DIFFERENT COMMUNICATION METHODS WITH IOT DEVICES (ZIGBEE, GSM, BLUETOOTH)</b>
<b>DATE:</b>	

**AIM:**

To Explore different communication methods with IoT devices (Zigbee, GSM, Bluetooth).

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Hardware & Software Requirement	Quantity
1	Bluetooth Module	1
2	Arduino UNO Board	1
3	Jump wires	Few
4	Arduino USB Cable	1
5	Serial Bluetooth Terminal	1

**THEORY**

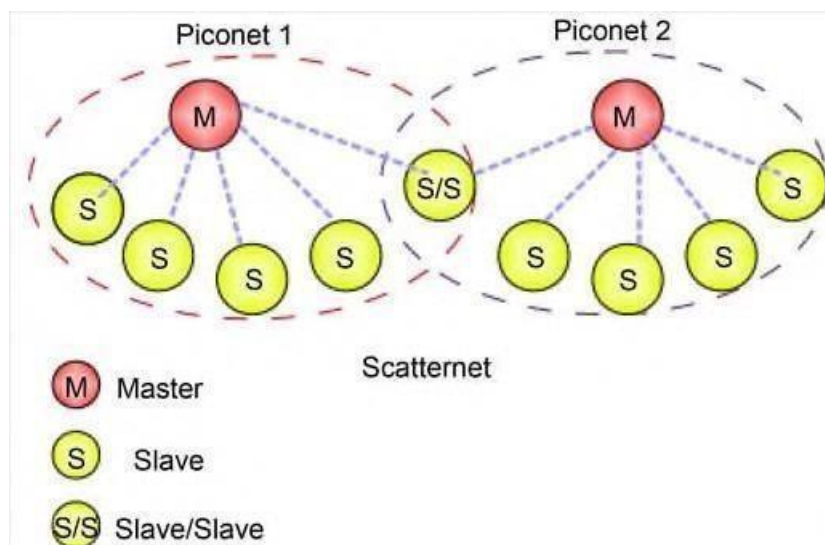
**Bluetooth:**

Bluetooth is a widely used short-range wireless communication technology that enables devices to exchange data over the 2.4 GHz ISM frequency band. Originally developed to eliminate the need for physical cables between devices such as phones, headsets, and computers, Bluetooth has become an essential part of many IoT applications. Its ability to provide reliable communication within a range of about 10 meters, combined with relatively low power consumption, makes it suitable for various consumer and industrial use cases.

A notable advancement in this technology is Bluetooth Low Energy (BLE), introduced in Bluetooth version 4.0. BLE is optimized for applications that require minimal power consumption and only periodic communication. This makes it especially well-suited for battery-powered IoT devices like fitness trackers, medical sensors, smartwatches, and other wearables. Unlike traditional Bluetooth, which

maintains continuous connections and higher energy usage, BLE operates in short bursts, conserving battery life while maintaining sufficient data transfer rates for typical IoT needs.

Bluetooth continues to play a critical role in the development of IoT solutions, particularly in areas such as healthcare monitoring, wearable devices, and smart home systems. It allows seamless interaction between mobile apps and IoT devices, enabling remote control and real-time data tracking. However, factors like its limited range, potential interference in crowded radio environments, and lower data throughput compared to alternatives like Wi-Fi must be considered when selecting it for specific applications. Still, its convenience, energy efficiency, and widespread device compatibility make Bluetooth a popular choice in the IoT ecosystem.



#### CONNECTIONS:

Arduino UNO Pin	Bluetooth Module	Arduino Development Board
VCC	5V	-
GND	GND	-
2	Tx	-
3	Rx	-
4	-	LED



**PROGRAM:**

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(3, 2); //HC-05 Tx & Rx is connected to Arduino 3 & 2 dpio

void setup(){

  Serial.begin(9600);

  mySerial.begin(9600);

  pinMode(13,OUTPUT);

  Serial.println("Initializing...");

  Serial.println("The device started, now you can pair it with bluetooth !");

}

void loop(){

  if(Serial.available()) {

    char data=Serial.read();

    Serial.print("Received: ");

    Serial.println(data);

    mySerial.write(data);

    if (data == '1') {

      digitalWrite(13, HIGH);

      Serial.println("LED ON");

    } else if (data == '0') {

      digitalWrite(13, LOW);

      Serial.println("LED OFF");

    }

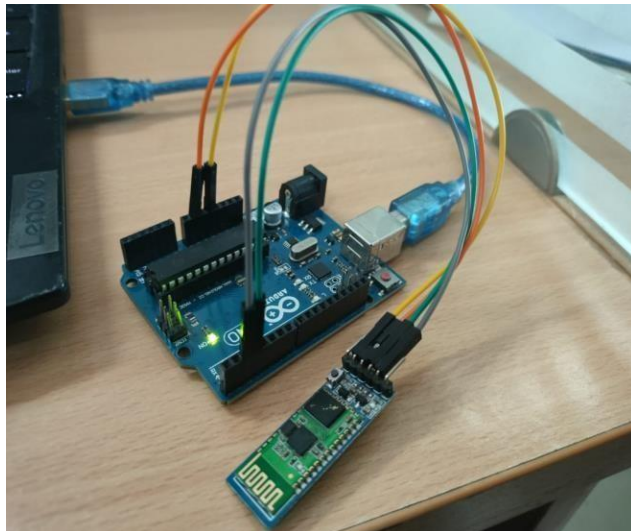
  }

  if(mySerial.available()) {

    char data = mySerial.read();
```

```
Serial.print("Received: ");  
  
Serial.println(data);  
  
if (data == '1') {  
    digitalWrite(13, HIGH);  
    Serial.println("LED ON");  
} else if (data == '0') {  
    digitalWrite(13, LOW);  
    Serial.println("LED OFF");  
}  
}}
```

## OUTPUT:



```
Received: 1  
LED ON  
Received:  
  
Received: 1  
Received: e  
Received: d  
Received:  
Received: o  
Received: n  
Received:  
Received:  
  
Received: 1  
Received: e  
Received: d  
Received:  
Received: o  
Received: f  
Received:  
  
Received: 0  
LED OFF  
Received:  
Received:  
  
Received: 1  
LED ON  
Received:  
Received:
```

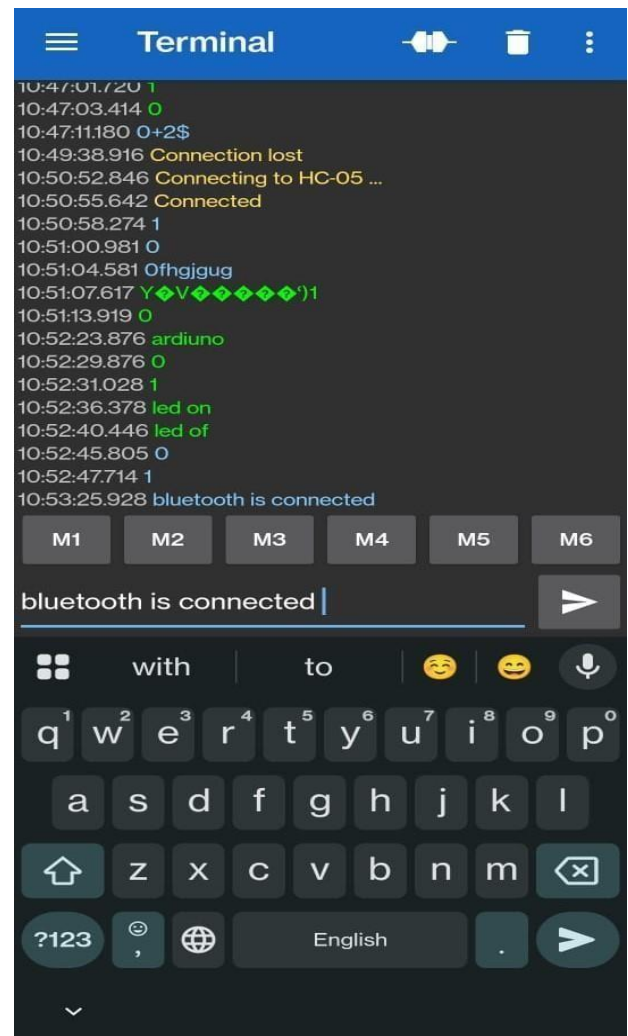


Terminal

```
10:41:33.854 Disconnected
10:41:34.438 Connecting to HC-05 ...
10:41:35.003 Connected
10:41:38.428 Disconnected
10:41:42.583 Connecting to HC-05 ...
10:41:44.638 Connected
10:42:06.356 hello
10:42:06.550 hello
10:42:11.044 jji
10:42:34.416 Connection lost
10:45:46.227 Connecting to HC-05 ...
10:45:50.083 Connected
10:46:15.777 hello
10:46:28.629 hello
10:46:38.417 hghfh
10:46:53.654 1
10:46:56.804 0
10:47:01.720 1
10:47:03.414 0
10:47:11.180 0+2$
10:49:38.916 Connection lost
10:50:52.846 Connecting to HC-05 ...
10:50:55.642 Connected
10:50:58.274 1
10:51:00.981 0
10:51:04.581 Ofhgjgug
10:51:07.617 Y V 1
10:51:13.919 0
10:52:23.876 arduino
10:52:29.876 0
10:52:31.028 1
10:52:36.378 led on
10:52:40.446 led of
10:52:45.805 0
10:52:47.714 1
10:53:25.928 bluetooth is connected
```

M1 M2 M3 M4 M5 M6

bluetooth is connected



Terminal

```
10:47:01.720 1
10:47:03.414 0
10:47:11.180 0+2$
10:49:38.916 Connection lost
10:50:52.846 Connecting to HC-05 ...
10:50:55.642 Connected
10:50:58.274 1
10:51:00.981 0
10:51:04.581 Ofhgjgug
10:51:07.617 Y V 1
10:51:13.919 0
10:52:23.876 arduino
10:52:29.876 0
10:52:31.028 1
10:52:36.378 led on
10:52:40.446 led of
10:52:45.805 0
10:52:47.714 1
10:53:25.928 bluetooth is connected
```

M1 M2 M3 M4 M5 M6

bluetooth is connected

with to

q w e r t y u i o p

a s d f g h j k l

z x c v b n m

?123 , . English

## RESULT:

Thus, the different communication methods with IoT devices (Zigbee, GSM, Bluetooth) are studied successfully.

<b>EXP NO.:</b> 07	<b>INTRODUCTION TO RASPBERRY PI PLATFORM AND PYTHON PROGRAMMING</b>
<b>DATE:</b>	

**AIM:**

To study the Raspberry Pi platform and python programming.

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Hardware & Software Requirement	Quantity
1	Raspberry Pi 3 Model	1
2	HDMI Cable for connect raspberry pi to monitor	1
3	Jump wires	Few
4	LED	1

**THEORY**

**Introduction to Raspberry Pi 5.3:**

Raspberry Pi 5.3 is the latest version of the small, affordable computer developed by the Raspberry Pi Foundation. It is a single-board computer, meaning all the main components like the CPU, RAM, USB ports, HDMI, Ethernet, etc., are built into a single board.

Raspberry Pi 5.3 offers a performance boost compared to earlier versions. It includes:

- Faster processor (Broadcom BCM2712 quad-core Cortex-A76)
- Improved GPU for video output and light gaming
- More RAM options (up to 8GB)
- Dual 4K display support
- PCIe 2.0 support for advanced hardware connections
- Better power management and cooling system

## PROCEDURE:

Step 1: Take your Raspberry Pi, 1 LED, and 2 jumper wires.

Step 2: Connect the long leg of the LED to GPIO 17 (Pin 11) using a jumper wire.

Step 3: Connect the short leg of the LED directly to GND (Pin 6) with another jumper wire.

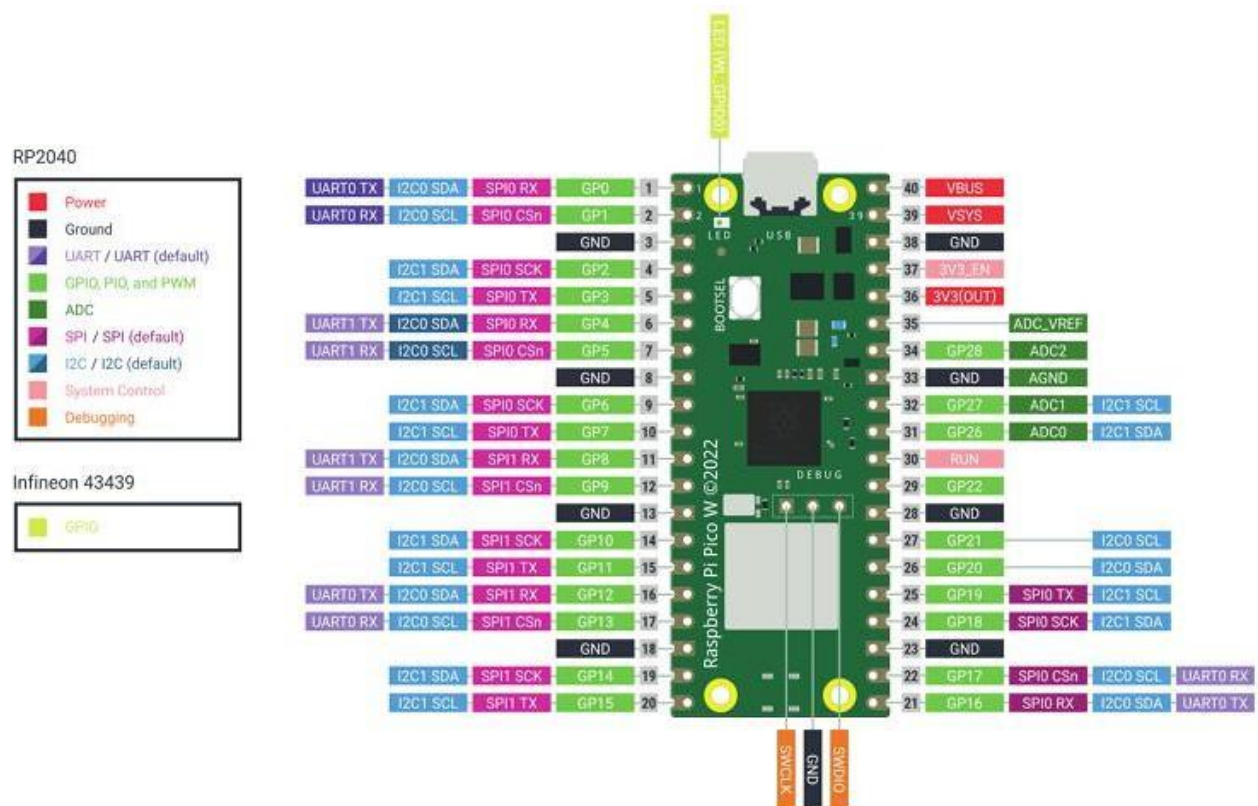
Step 4: Plug in HDMI, keyboard, and mouse, then power on your Pi.

Step 5: Open terminal and install GPIO library `sudo apt install python3-rpi.gpio`.

Step 6: Create a Python file and write a simple code to turn GPIO 17 ON and OFF.

Step 7: Run the program using `python3 blink.py` and watch the LED blink.

## PIN DIAGRAM:



**PROGRAM:**

```
import RPi.GPIO as GPIO

from time import sleep

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(7, GPIO.OUT, initial=GPIO.LOW)

while True:

    GPIO.output(7, GPIO.HIGH)

    print("LED ON")

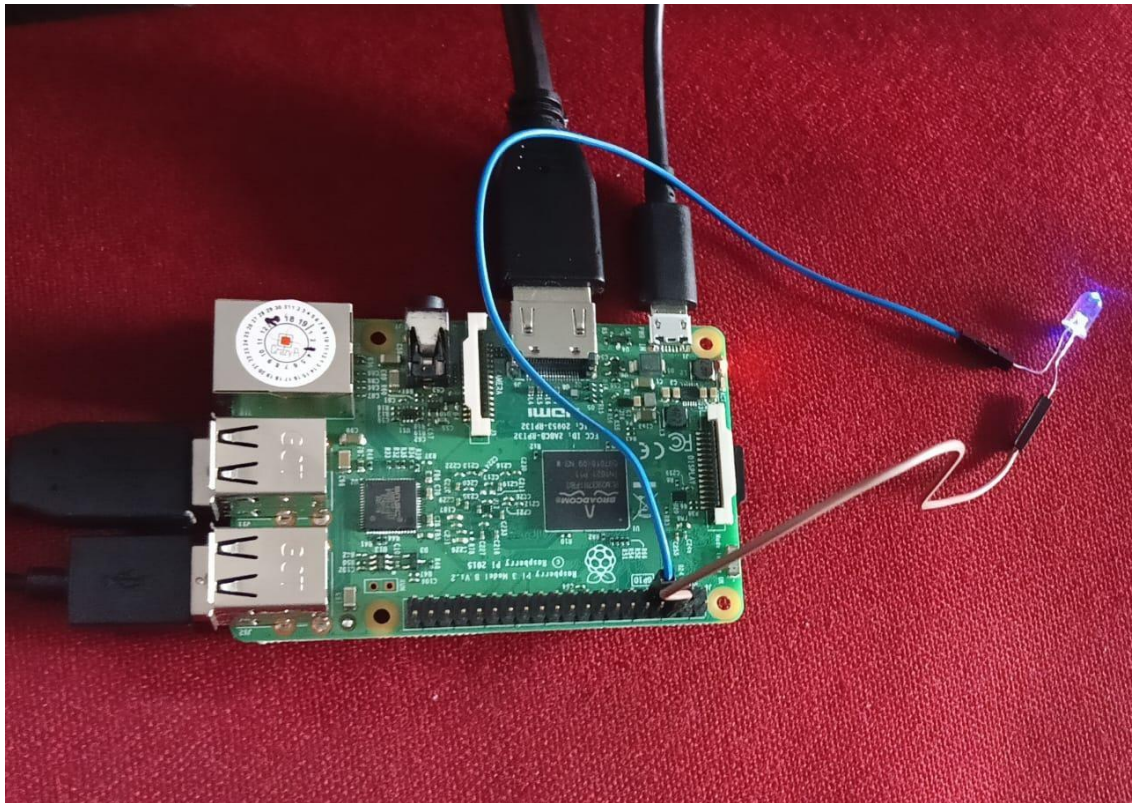
    sleep(1)

    GPIO.output(7, GPIO.LOW)

    print("LED OFF")

    sleep(1)
```

## OUTPUT:



## RESULT:

Thus, the Raspberry Pi Platform and python programming are studied successfully.

<b>EXP NO.:</b> 08	<b>INTERFACING SENSORS WITH RASPBERRY PI</b>
<b>DATE:</b>	

**AIM:**

To interface the IR sensor and Ultrasonic sensor with Raspberry Pico.

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Hardware & Software Requirement	Quantity
1	Thonny IDE	1
2	Raspberry Pi 3	1
3	Jump wires	Few
4	Micro USB Cable	1
5	IR Sensor	1

**PROCEDURE:**

Step 1: Connect a sensor to Raspberry Pi and write Python code to read the data.

Step 2: Create a ThingSpeak account, make a channel, and note the Write API Key.

Step 3: Install the requests library on Raspberry Pi using `sudo pip install requests`.

Step 4: Write Python code to send sensor data to ThingSpeak using the API Key.

Step 5: Run the code and check if data shows up on your ThingSpeak channel.



**PROGRAM (IR SENSOR):**

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
IR_PIN = 17
GPIO.setup(IR_PIN, GPIO.IN)
print("IR Sensor Test - Press Ctrl+C to exit")
try:
    while True:
        if GPIO.input(IR_PIN) == 0:
            print("Obstacle Detected!")
        else:
            print("No Obstacle")
            time.sleep(0.5)
except KeyboardInterrupt:
    print("Exiting Program")
finally:
    GPIO.cleanup()
```

## OUTPUT:



```
Shell %  
No Obstacle  
No Obstacle  
No Obstacle  
No Obstacle  
No Obstacle  
No Obstacle  
No Obstacle  
Obstacle Detected!  
Obstacle Detected!
```

## RESULT:

Thus, the IR sensor and Ultrasound sensor are interfaced with Raspberry Pi executed successfully.

<b>EXP NO.:</b> 09	<b>COMMUNICATE BETWEEN ARDUINO AND RASPBERRY PI USING ANY WIRELESS MEDIUM</b>
<b>DATE:</b>	

**AIM:**

To study the program to Communicate between Arduino and Raspberry PI using any wireless medium (Bluetooth)

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Hardware & Software Requirement	Quantity
1	Thonny IDE	1
2	Raspberry Pi Pico Development Board	1
3	Jump wires	Few
4	Micro USB Cable	1
5	Arduino UNO Board	1

**PROCEDURE:**

Step 1: Connect Bluetooth modules to Arduino and Raspberry Pi Pico.

Step 2: Arduino sends 'A' and 'B' one after another every second.

Step 3: Raspberry Pi Pico receives data using UART.

Step 4: Check if received data is 'A' or 'B'.

Step 5: If 'A', turn LED ON; if 'B', turn LED OFF.

Step 6: Confirm LED status and data is received correctly.

## PROGRAM:

### MASTER- ARDUINO:

```
#include SoftwareSerial mySerial(2,3); //rx,tx

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

void loop() {
  mySerial.write('A');
  delay(1000);
  mySerial.write('B');
  delay(1000);
}
```

### CONNECTIONS:

Arduino UNO Pin	Arduino Development Board	Bluetooth Module
<b>2</b>	-	<b>Tx</b>
<b>3</b>	-	<b>Rx</b>
-	<b>GND</b>	<b>GND</b>
-	<b>5V</b>	<b>5V</b>

### SLAVE - RASPBERRY PI PICO

```
from machine import Pin, UART

uart = UART(0, 9600)

led = Pin(16, Pin.OUT)

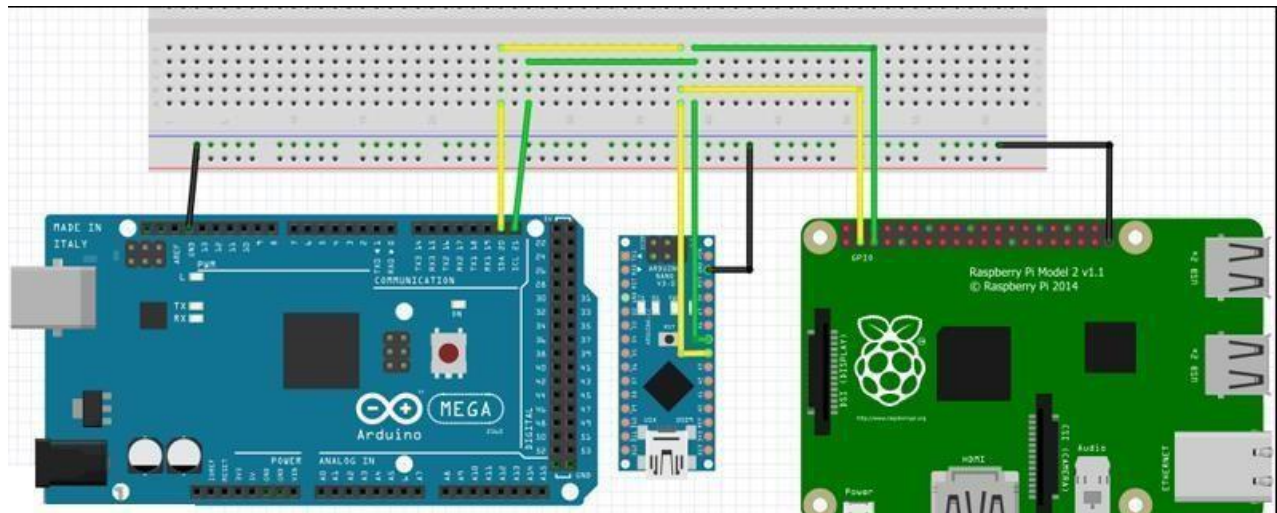
while True:
  if uart.any() > 0:
    data = uart.read()
    print(data)
    if "A" in data:
      led.value(1)
      print('LED on \n')
      uart.write('LED on \n')
    elif "B" in data:
```

```
led.value(0)
print('LED off \n')
uart.write('LED off \n')
```

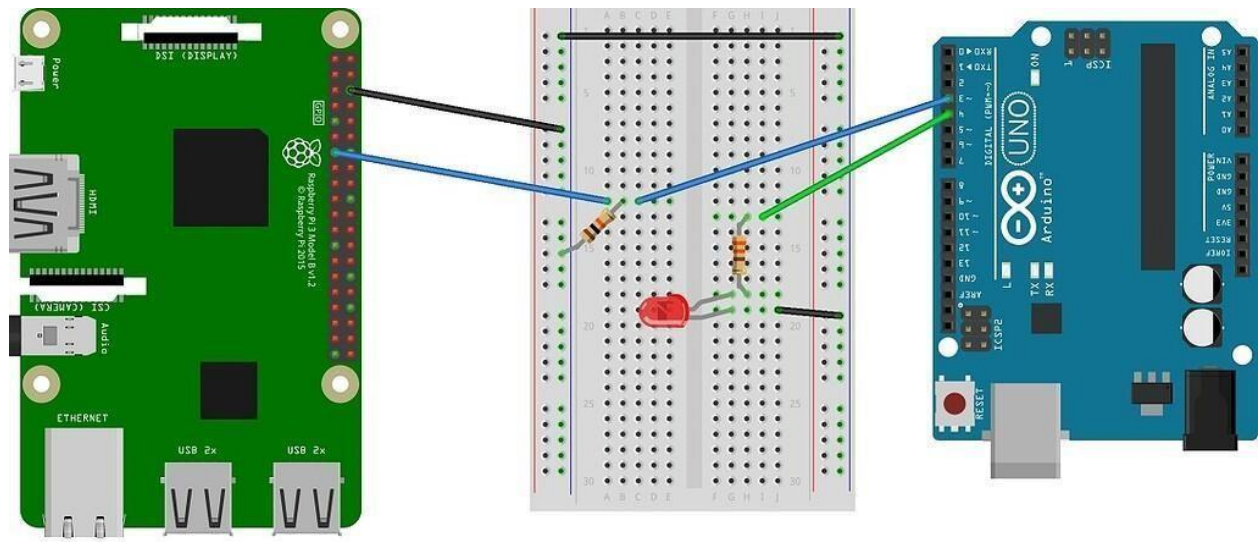
### CONNECTIONS:

Raspberry Pi Pico Pin	Raspberry Pi Pico Development Board	Bluetooth Module
GP16	LED	-
VCC	-	+5V
GND	-	GND
GP1	-	Tx
GP0	-	Rx

### OUTPUT:



## INTERFACING FOR LED:



## RESULT:

Thus the program to Communicate between Arduino and Raspberry PI using any wireless medium (Bluetooth) was written and executed successfully.

<b>EXP NO.:</b> 10	<b>SETUP A CLOUD PLATFORM TO LOG THE DATA</b>
<b>DATE:</b>	

**AIM:**

To set up a cloud platform to log the data from IoT devices.

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Software Requirement	Quantity
1	Thingspeak	1

**CLOUD PLATFORM – THINGSPEAK**

ThingSpeak is an open-source Internet of Things (IoT) cloud platform that enables users to collect, store, analyze, and visualize sensor data in real time. Developed by MathWorks, ThingSpeak integrates seamlessly with MATLAB for advanced analytics and predictive modeling, making it a powerful tool for engineers, researchers, and developers working on IoT applications. It allows users to set up channels to collect data from sensors or devices via HTTP or MQTT protocols, and it supports the creation of visualizations such as line graphs, bar charts, and maps directly within the platform. With built-in features like data aggregation, event scheduling, and alerting (via email or Twitter), ThingSpeak facilitates rapid prototyping and deployment of IoT systems without the need to build a backend from scratch.

In addition to its real-time data handling capabilities, ThingSpeak is widely appreciated for its accessibility and flexibility. It offers both free and paid versions, where the free tier is ideal for academic and hobbyist projects, and the paid licenses support higher data rates and commercial usage. The platform also supports integration with third-party services and devices such as Arduino, Raspberry Pi, and ESP8266, making it highly versatile for a broad range of IoT projects. Its MATLAB analytics integration allows users to process and analyze data directly on the cloud, enabling applications like predictive maintenance, anomaly detection, and machine learning-driven insights. Overall, ThingSpeak stands out as a comprehensive and user-friendly IoT cloud solution that bridges hardware, data analytics, and visualization effectively.

## PROCEDURE:

Step 1: go to thingspeak.com and click sign up to create a free mathworks account by entering your email, name, and password, then verify your email to activate the account.

Step 2: after activating the account, log in to thingspeak.com using your mathworks credentials to access the main dashboard.

Step 3: click on channels, select my channels, and then click new channel to begin creating your own channel.

Step 4: enter a name for your channel, optionally add a description, enable the fields you need to store sensor data, and scroll down to click save channel.

Step 5: once the channel is created, note the channel id and api keys which you will use to send or retrieve data, and now you can start sending data from your device and visualizing it with charts on thingspeak.

## OUTPUT:

The screenshot shows the 'My Channels' page on the ThingSpeak website. The header includes the ThingSpeak logo and navigation links for Channels, Apps, Devices, and Support. A 'New Channel' button is prominently displayed. A search bar with the placeholder 'Search by tag' is also present. On the right side, there is a 'Help' section with instructions on how to collect data and create a new channel, followed by 'Examples' listing Arduino, ESP8266, Raspberry Pi, and Netduino Plus. An 'Upgrade' section is also visible. A cookie consent banner is at the bottom.

The screenshot shows the 'New Channel' creation form. It includes fields for 'Name' and 'Description'. Below these are eight 'Field' checkboxes, each with a corresponding 'Field Label' input. The 'Field 1' checkbox is checked. To the right, there is a 'Help' section titled 'Channel Settings' with detailed instructions for each field, including 'Percentage complete', 'Channel Name', 'Description', 'Fields', 'Metadata', 'Tags', 'Link to External Site', and 'Show Channel Location'. A cookie consent banner is at the bottom.



ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy

## IR 2

Channel ID: 2956965  
Author: mrua000037786096  
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

### Write API Key

Key: TRHWFS3K4LLTD69E

Generate New Write API Key

### Read API Keys

Key: DQ5IPJUBFTJGE8T5

Note:

Save Note Cancel API Key

### Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

### API Keys Settings

- Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click [Generate New Write API Key](#).
- Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click [Generate New Read API Key](#) to generate an additional read key for the channel.
- Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

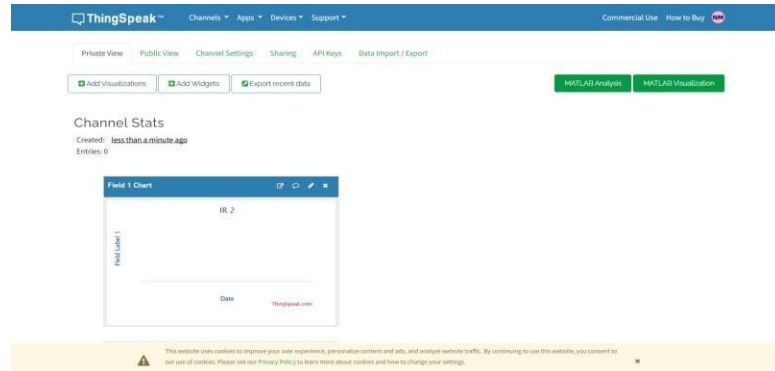
### API Requests

Write a Channel Feed

GET [https://api.thingspeak.com/channels/api\\_key/TRHWFS3K4LLTD69E/feeds/1](https://api.thingspeak.com/channels/api_key/TRHWFS3K4LLTD69E/feeds/1)

Read a Channel Feed

This website uses cookies to improve your user experience, personalize content and ads, and analyze website usage. By continuing to use this website, you consent to our use of cookies. Please see our [Privacy Policy](#) to learn more about cookies and how to change your settings.



## RESULT:

Thus, the cloud program to log the data from IoT devices is set up successfully.

<b>EXP NO.:</b> 11	<b>LOG DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM</b>
<b>DATE:</b>	

**AIM:**

To write and execute the program Log Data using Raspberry PI and upload it to the cloud platform

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Hardware & Software Requirement	Quantity
1	Thonny IDE	1
2	Raspberry Pi Pico Development Board	1
3	Jump wires	Few
4	Micro USB Cable	1

**PROCEDURE:**

step 1: connect the ir sensor to raspberry pi pin 17 and install needed libraries.

step 2: set the pin as input and check if obstacle is detected or not.

step 3: get the thingspeak api key ready and set value 1 for obstacle, 0 for no obstacle.

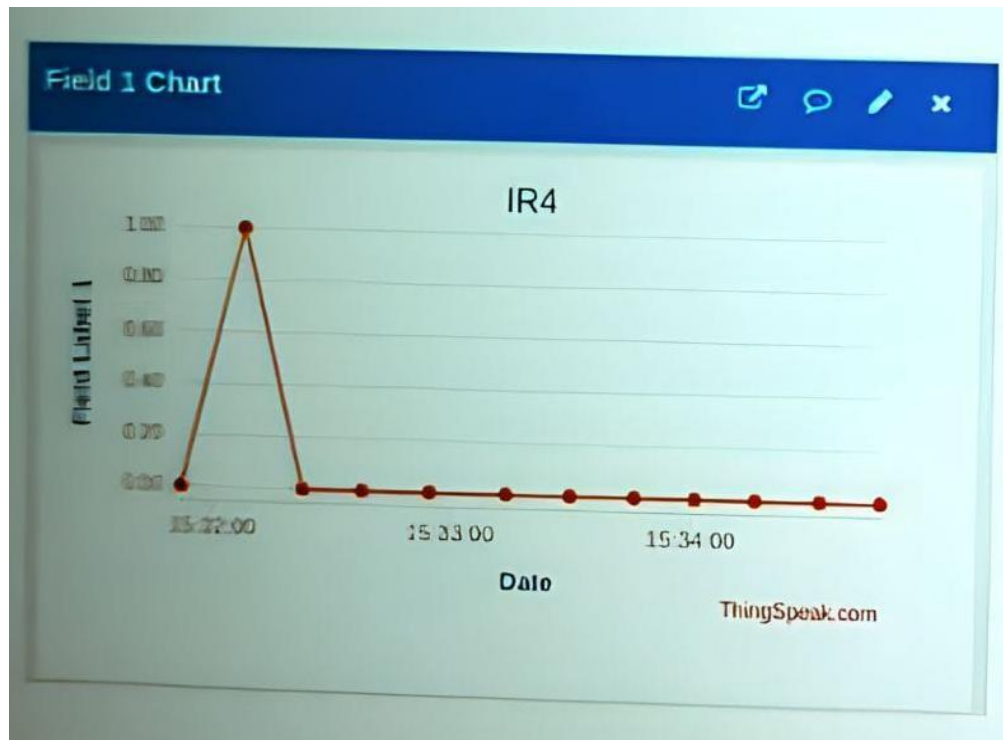
step 4: send the value to thingspeak using a get request.

step 5: keep repeating and clean up gpio when you stop the program.

**PROGRAM:**

```
import RPi.GPIO as GPIO
import time
import requests
API_KEY = "Z7FKADF66448MH"
URL = f"https://api.thingspeak.com/update"
GPIO.setmode(GPIO.BCM)
IR_PIN = 17
GPIO.setup(IR_PIN, GPIO.IN)
print("IR Sensor Test - Sending data to Thingspeak. Press Ctrl+C to exit")
try:
    while True:
        if GPIO.input(IR_PIN) == 0:
            obstacle = 1
            print("Obstacle Detected!")
        else:
            obstacle = 0
            print("No Obstacle")
        payload = {'api_key': API_KEY, 'field1': obstacle}
        response = requests.get(URL, params=payload)
        if response.status_code == 200:
            print("Data sent to Thingspeak.")
        else:
            print(f"Failed to send data. HTTP {response.status_code}")
            time.sleep(0.5)
except KeyboardInterrupt:
    print("Exiting Program")
finally:
    GPIO.cleanup()
```

## OUTPUT:



```
Shell x
>>> %Run 8.py
IR Sensor Test - Sending data to ThingSpeak. Press Ctrl+C to exit
No Obstacle
Data sent to ThingSpeak.
Obstacle Detected!
Data sent to ThingSpeak.
No Obstacle
Data sent to ThingSpeak.
No Obstacle
Data sent to ThingSpeak.
No Obstacle
Data sent to ThingSpeak.
No Obstacle
Data sent to ThingSpeak.
No Obstacle
Data sent to ThingSpeak.
No Obstacle
Data sent to ThingSpeak.
No Obstacle
```

## RESULT:

Thus, the program to execute Log Data using Raspberry pi and uploading it to the cloud platform is done successfully.

<b>EXP NO.:</b> 12	<b>DESIGN AN IOT BASED SYSTEM</b>
<b>DATE:</b>	

**AIM:**

To write a program using sensors for carparking assist.

**HARDWARE & SOFTWARE TOOLS REQUIRED:**

S.NO	Hardware & Software Requirement	Quantity
1	PC with windows	1
2	Arduino 1.6.5	1
3	Proteus 8.0 with Sensor Library	1

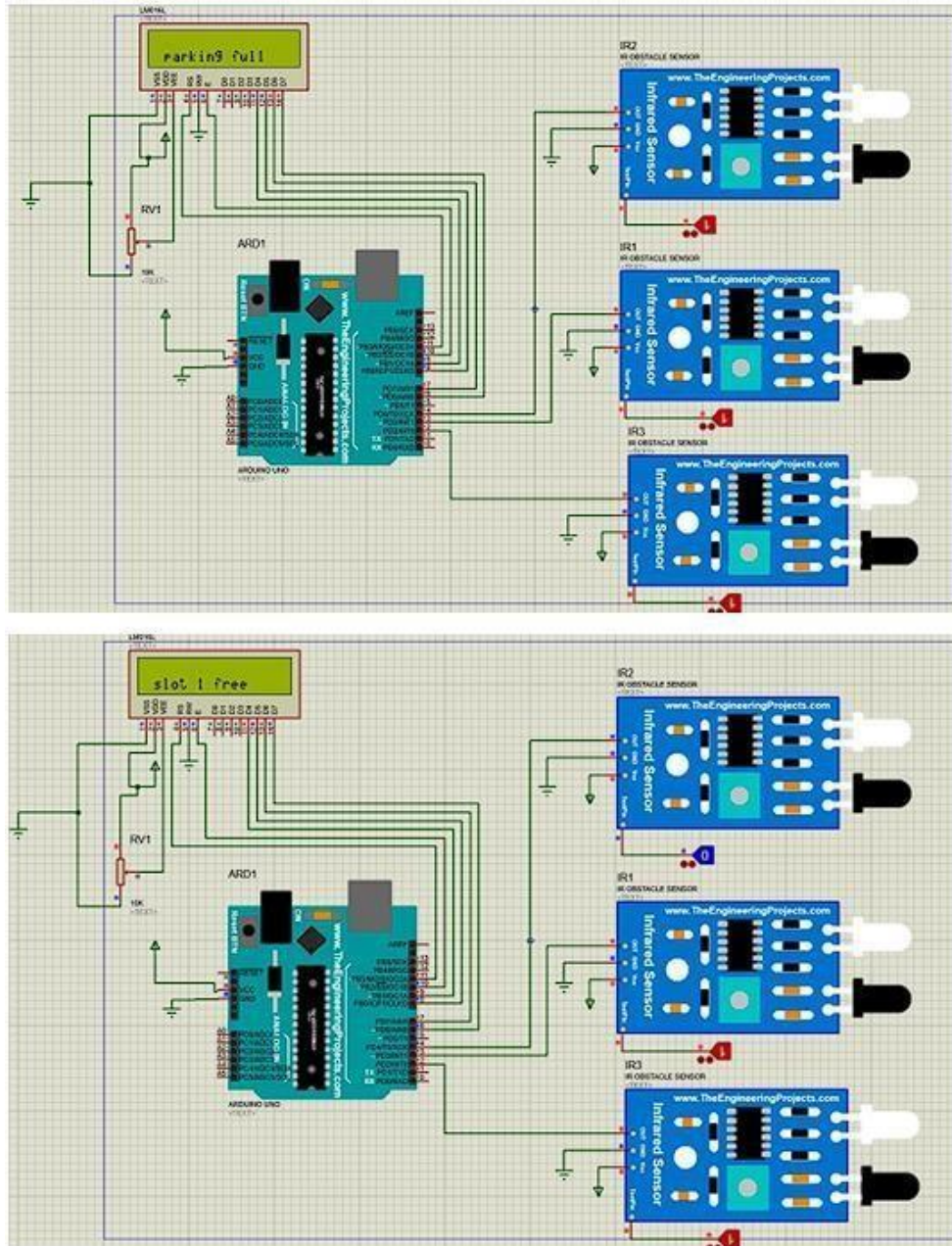
**THEORY :**

The infrared Obstacle Sensor Module has a built-in IR transmitter and IR receiver that sends out IR energy and looks for reflected IR energy to detect the presence of any obstacle in front of the sensor module. IR sensor is an electronic device, that emits the light in order to sense some object of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations. The program will give feedback to the driver based on the proximity of the car to the obstacle. We'll use an LED to indicate the presence of Free slot. The usage of this experiment becomes the major part in numerous and exclusive Malls, theatres, and Big Residencies.

**PROGRAM:**

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(11, 10, 9, 8, 7, 6);
int sen1 = 4;
int sen2 = 3;
int sen3 = 2;
void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  pinMode(sen1, INPUT);
  pinMode(sen2, INPUT);
  pinMode(sen3, INPUT);
}
void loop() {
  int irValue1 = digitalRead(sen1);
  int irValue2 = digitalRead(sen2);
  int irValue3 = digitalRead(sen3);
  if (irValue1 == LOW && irValue2 == HIGH && irValue3 == HIGH) {
    lcd.clear();
    lcd.setCursor(1, 1);
    lcd.print("slot 1 free");
    delay(100);
  }
  else if (irValue2 == LOW && irValue1 == HIGH && irValue3 == HIGH) {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("slot 2 free");
    delay(100);
  }
  else if (irValue3 == LOW && irValue1 == HIGH && irValue2 == HIGH) {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("slot 3 free");
    delay(100);
  }
  else if (irValue1 == LOW || irValue2 == LOW || irValue3 == LOW) {
    lcd.clear();
    lcd.setCursor(1, 1);
    lcd.print("welcome");
    delay(100);
  }
  else if (irValue1 == HIGH && irValue2 == HIGH && irValue3 == HIGH) {
    lcd.clear();
    lcd.setCursor(1, 1);
    lcd.print("parking full");
    delay(100);
  }
}
```

## OUTPUT:



## RESULT:

Thus, the program for Carparking assist using IR Sensors was designed successfully.

