EXP NO. : 01	**8051 ASSEMBLY LANGUAGE EXPERIMENT USING SIMULATOR**
DATE:	

AIM:

To write an 8051 Assembly language experiment using simulator.

APPARATUS REQUIRED

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

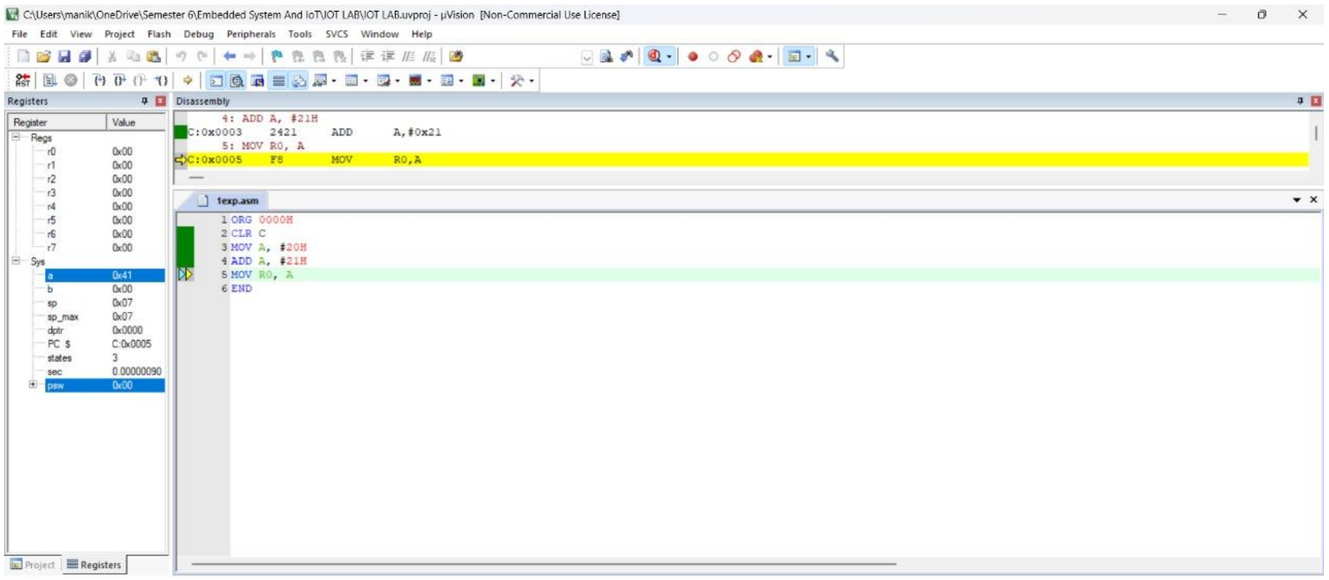
PROCEDURE:

Step 1 :	Create a new project in Keil µ 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.

PROGRAM:

```
ORG

0000H

CLR C

MOV A, #20H

ADD A, #21H

MOV R0, A

END
```

OUTPUT:


C:\Users\manik\OneDrive\Semester 6\Embedded System And IoT\IOT LAB\IOT LAB.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
1exp.asm
Project
Registers

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x41
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x0005
states	3
sec	0.00000090
psw	0x00

```
     4: ADD A, #21H
C:0x0003    2421     ADD      A,#0x21
     5: MOV R0, A
C:0x0005    F8       MOV      R0,A
```

```
ORG 0000H
CLR C
MOV A, #20H
ADD A, #21H
MOV R0, A
END
```

RESULT:

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

EXP NO. : 02	**TEST DATA TRANSFER BETWEEN REGISTERS AND MEMORY**
DATE:	

AIM:

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

APPARATUS REQUIRED

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

PROCEDURE:

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

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

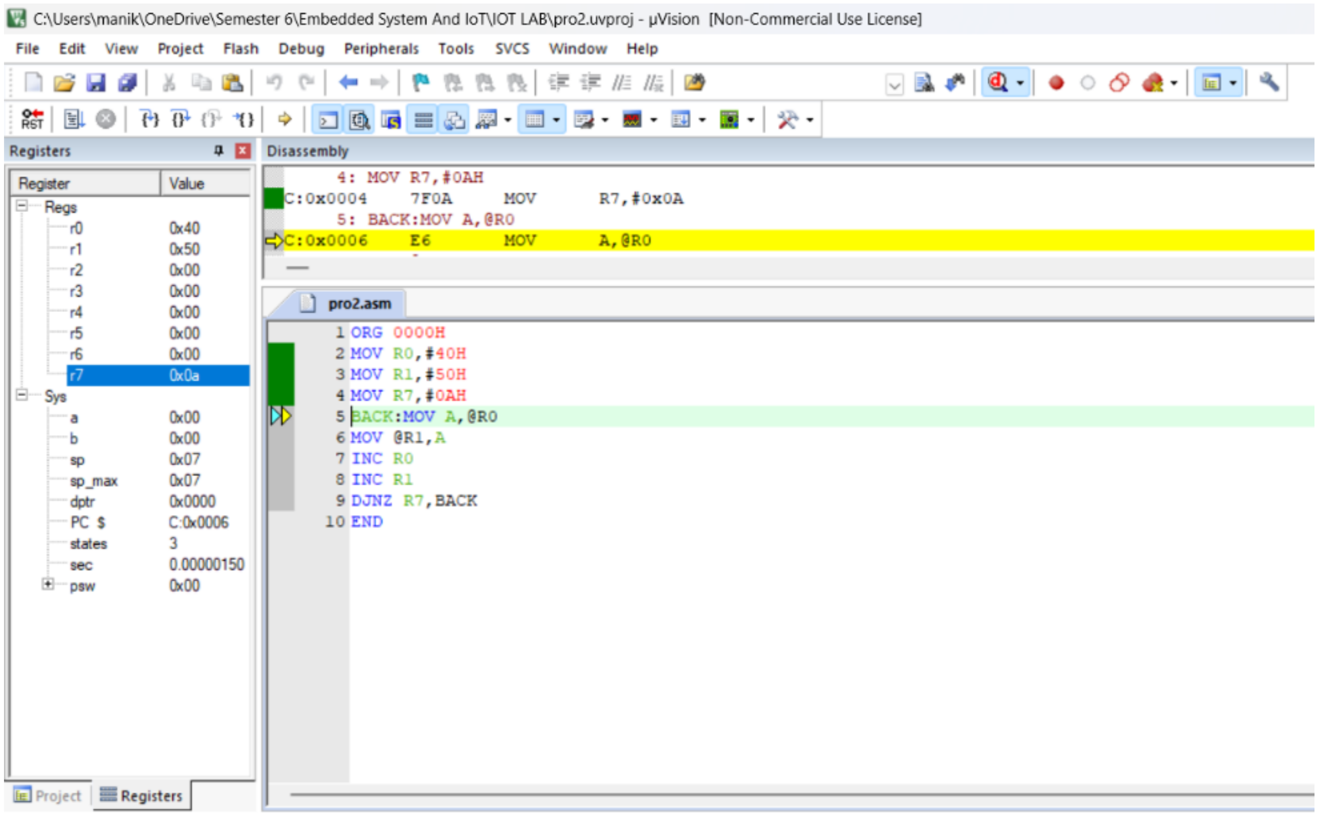
Step 3 : Debug the Code using Debug → Start / Stop Debug Session.

Step 4 : Stop the Program.

PROGRAM:

```
ORG 0000H

CLR C

MOV R0, #30H

MOV R1, #40H

MOV R7, #06H

BACK: MOV A, @R0

MOV @R1, A

INC R0

INC R1

DJNZ R7, BACK

END
```

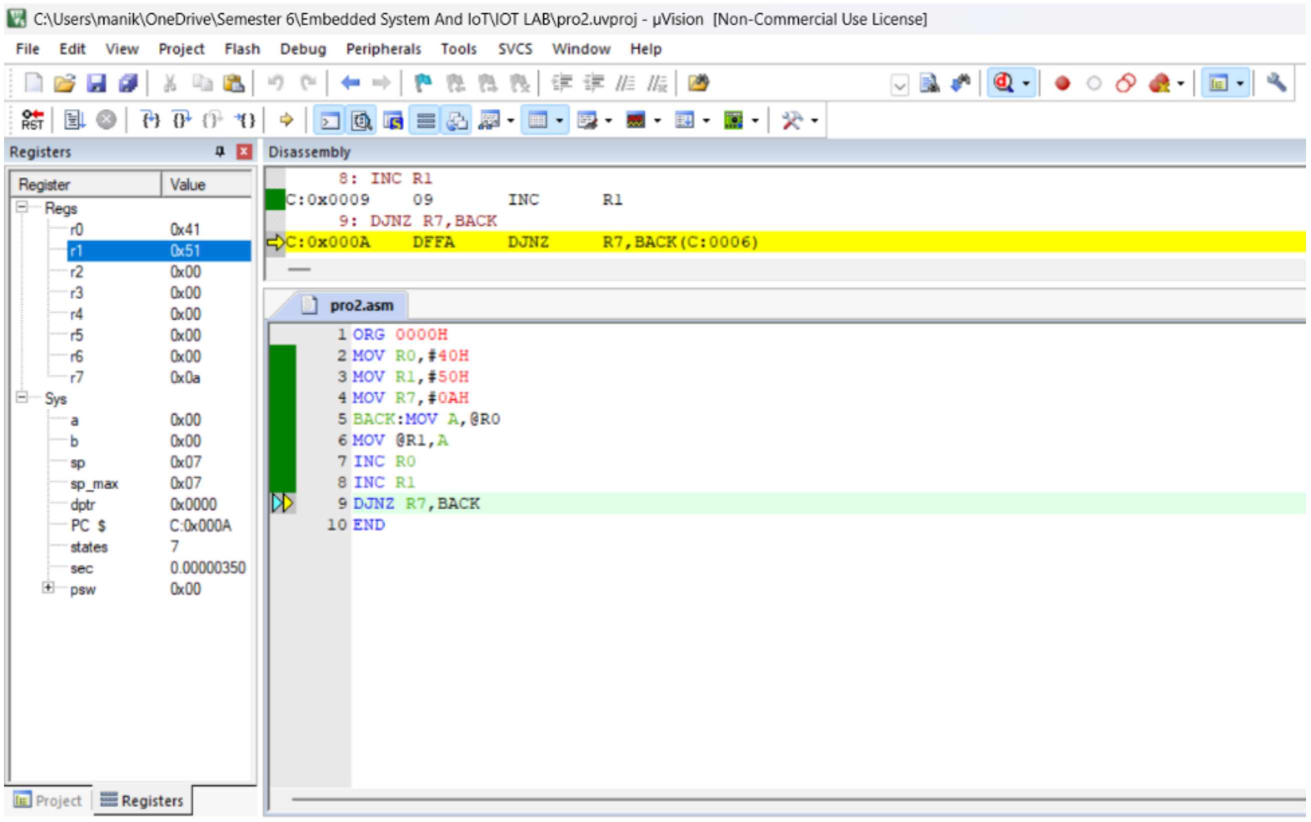
OUTPUT:


C:\Users\manik\OneDrive\Semester 6\Embedded System And IoT\IOT LAB\pro2.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
pro2.asm
Project
Registers

Register	Value
Regs	
r0	0x40
r1	0x50
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x0a
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x0006
states	3
sec	0.00000150
psw	0x00

```
     4: MOV R7,#0AH
C:0x0004    7F0A     MOV      R7,#0x0A
     5: BACK:MOV A,@R0
C:0x0006    E6       MOV      A,@R0
```

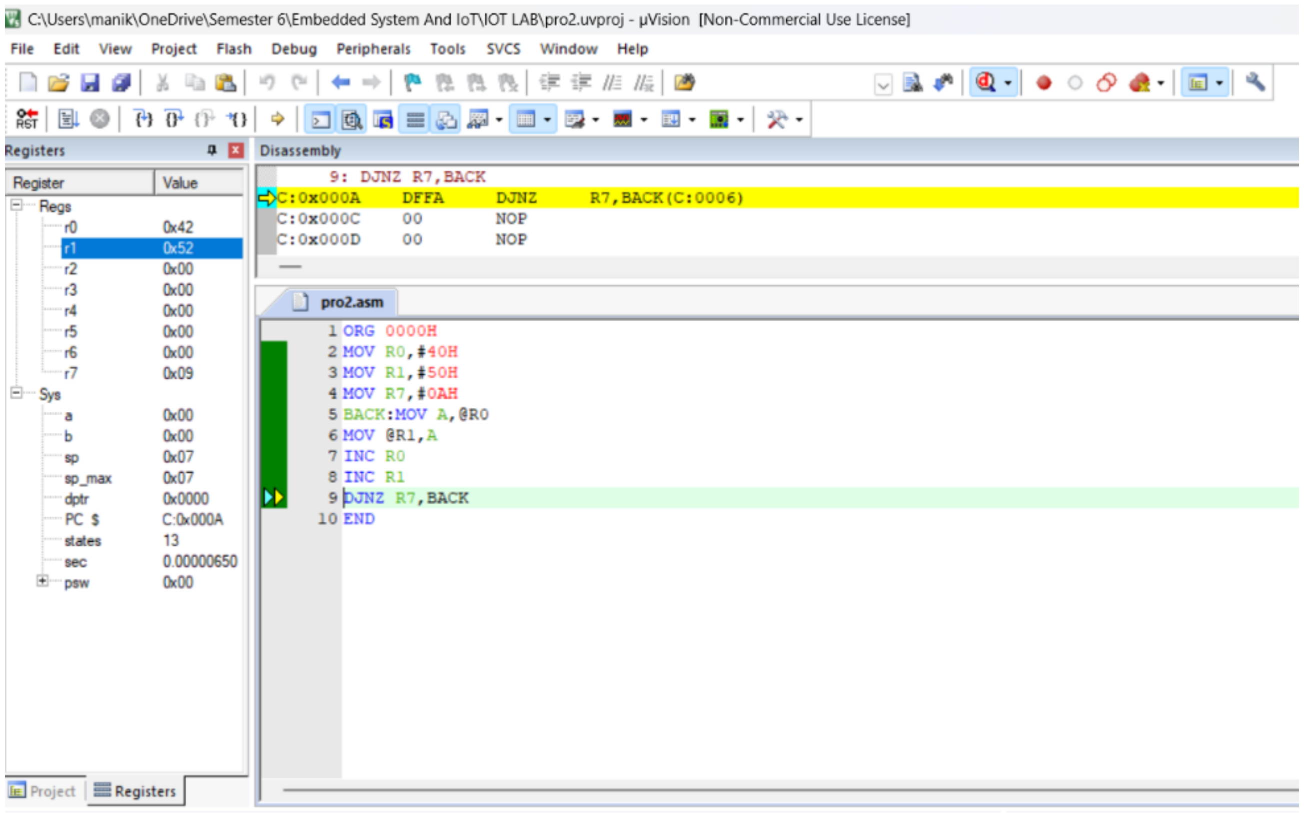
```
ORG 0000H
MOV R0,#40H
MOV R1,#50H
MOV R7,#0AH
BACK:MOV A,@R0
MOV @R1,A
INC R0
INC R1
DJNZ R7,BACK
END
```


C:\Users\manik\OneDrive\Semester 6\Embedded System And IoT\IOT LAB\pro2.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
pro2.asm
Project
Registers

Register	Value
Regs	
r0	0x41
r1	0x51
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x0a
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x000A
states	7
sec	0.00000350
psw	0x00

```
     8: INC R1
C:0x0009    09       INC      R1
     9: DJNZ R7,BACK
C:0x000A    DFFA     DJNZ     R7,BACK(C:0006)
```

```
ORG 0000H
MOV R0,#40H
MOV R1,#50H
MOV R7,#0AH
BACK:MOV A,@R0
MOV @R1,A
INC R0
INC R1
DJNZ R7,BACK
END
```


C:\Users\manik\OneDrive\Semester 6\Embedded System And IoT\IOT LAB\pro2.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
pro2.asm
Project
Registers

Register	Value
Regs	
r0	0x42
r1	0x52
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x09
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x000A
states	13
sec	0.00000650
psw	0x00

```
     9: DJNZ R7,BACK
C:0x000A    DFFA      DJNZ     R7,BACK(C:0006)
C:0x000C    00        NOP
C:0x000D    00        NOP
```

```
ORG 0000H
MOV R0,#40H
MOV R1,#50H
MOV R7,#0AH
BACK:MOV A,@R0
MOV @R1,A
INC R0
INC R1
DJNZ R7,BACK
END
```

RESULT:

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

EXP NO. : 03	**ALU PROGRAM**
DATE:	

AIM:

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

APPARATUS REQUIRED

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

PROCEDURE:

Step 1 : Create a new project in Keil µ 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.

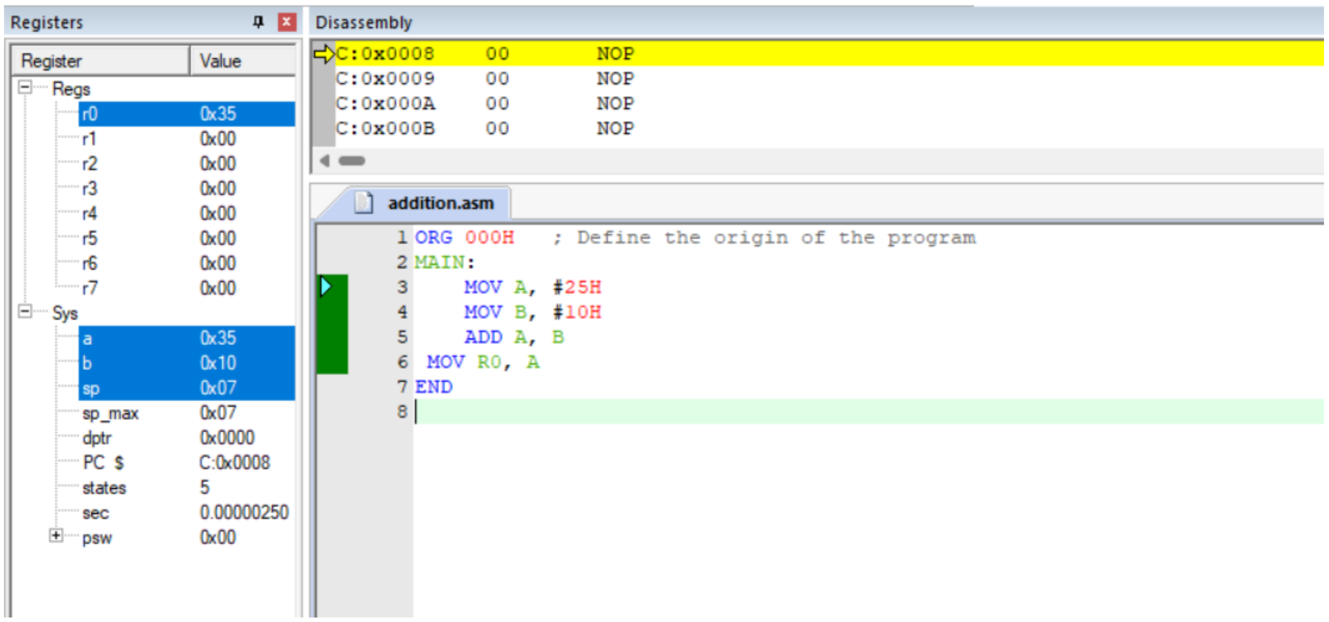
Typically, the ALU has direct input and output access to the processor controller, main memory (random access memory or RAM in a personal computer) and input/output devices. Inputs and outputs flow along an electronic path that is called a bus. The input consists of an instruction word, sometimes called 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 PROGRAM:

```
ORG 000H   ; Define the origin of the program

MAIN:

   MOV A, #25H

   MOV B, #10H

   ADD A, B

 MOV R0, A

END
```

OUTPUT:



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

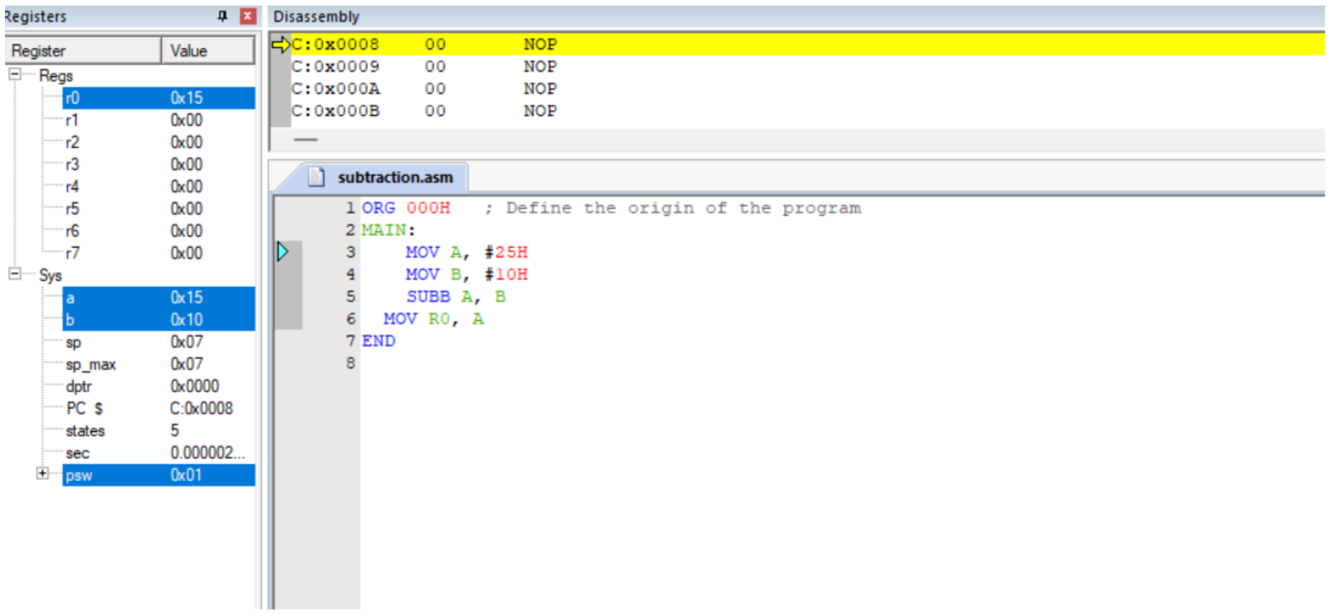
```
C:0x0008    00    NOP
C:0x0009    00    NOP
C:0x000A    00    NOP
C:0x000B    00    NOP
```

```
ORG 000H    ; Define the origin of the program
MAIN:
    MOV A, #25H
    MOV B, #10H
    ADD A, B
 MOV R0, A
END
```

SUBTRACTION PROGRAM:

```
ORG 000H   ; Define the origin of the program

MAIN:

    MOV A, #25H

    MOV B, #10H

    SUBB A, B

  MOV R0, A

END
```

OUTPUT:



```
Registers
Register    Value
Regs
  r0        0x15
  r1        0x00
  r2        0x00
  r3        0x00
  r4        0x00
  r5        0x00
  r6        0x00
  r7        0x00
Sys
  a         0x15
  b         0x10
  sp        0x07
  sp_max    0x07
  dptr      0x0000
  PC $      C:0x0008
  states    5
  sec       0.000002...
  psw       0x01
```

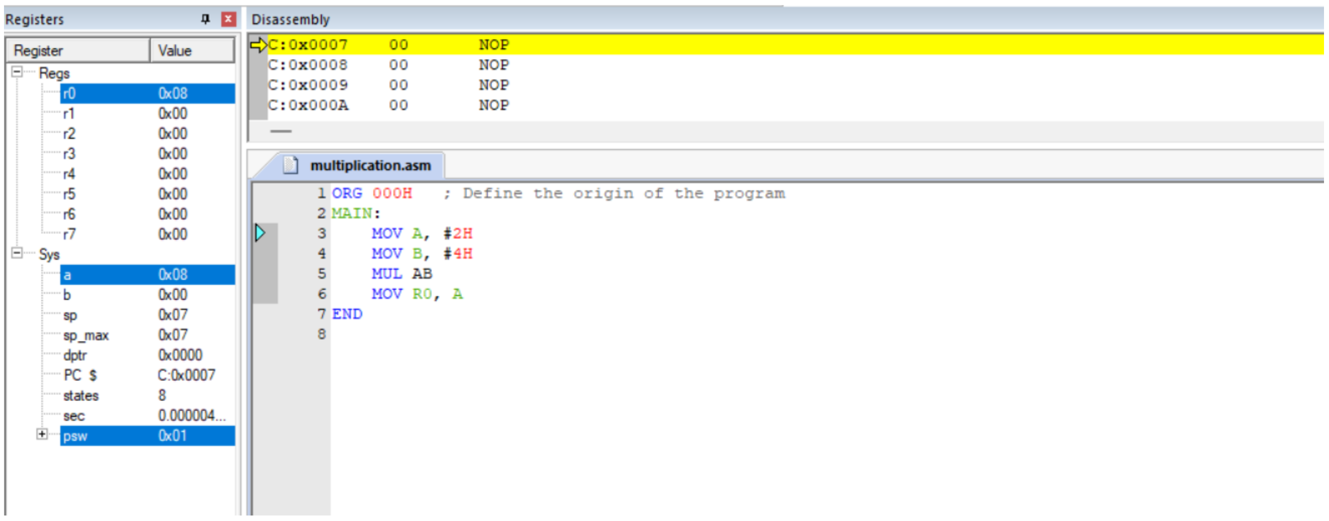
```
Disassembly
C:0x0008    00    NOP
C:0x0009    00    NOP
C:0x000A    00    NOP
C:0x000B    00    NOP
```

```
subtraction.asm
ORG 000H   ; Define the origin of the program
MAIN:
    MOV A, #25H
    MOV B, #10H
    SUBB A, B
  MOV R0, A
END
```

MULTIPLICATION PROGRAM:

```
ORG 000H   ; Define the origin of the program
MAIN:
   MOV A, #2H
   MOV B, #4H
   MUL AB
   MOV R0, A
END
```

OUTPUT:



```
Registers
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       0.000004...
  psw       0x01
```

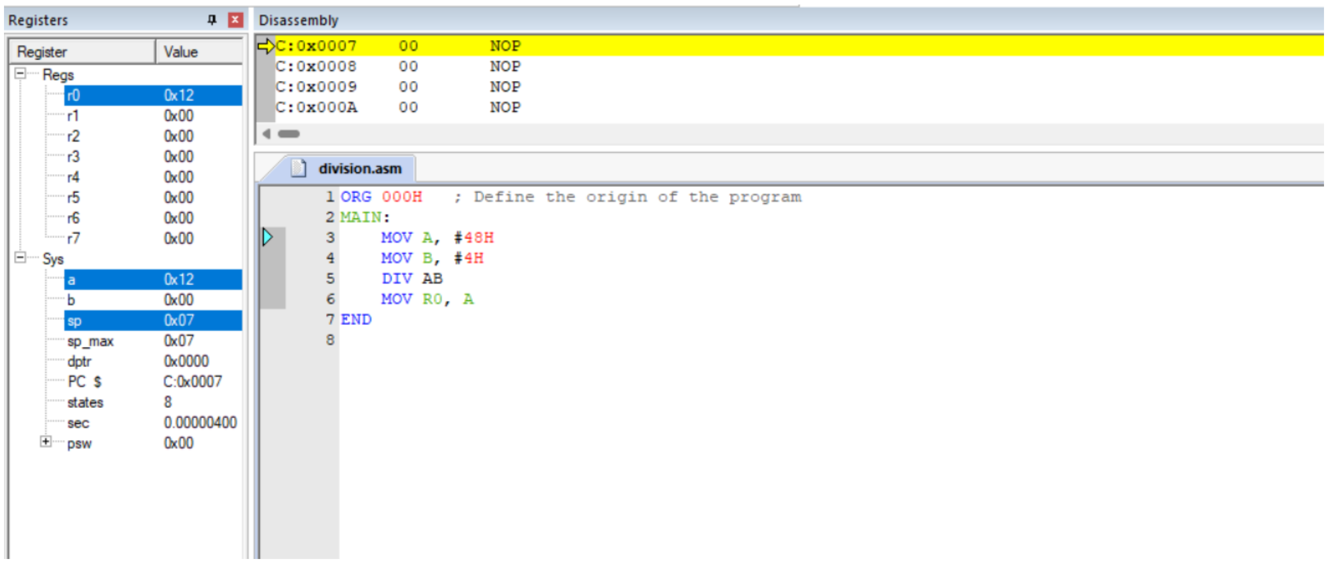
```
Disassembly
C:0x0007    00    NOP
C:0x0008    00    NOP
C:0x0009    00    NOP
C:0x000A    00    NOP
```

```
multiplication.asm
ORG 000H   ; Define the origin of the program
MAIN:
    MOV A, #2H
    MOV B, #4H
    MUL AB
    MOV R0, A
END
```

DIVISION PROGRAM:

```
ORG 000H   ; Define the origin of the program

MAIN:

   MOV A, #48H

   MOV B, #4H

   DIV AB

   MOV R0, A

END
```

OUTPUT:



Registers

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

Disassembly

```
C:0x0007    00        NOP
C:0x0008    00        NOP
C:0x0009    00        NOP
C:0x000A    00        NOP
```

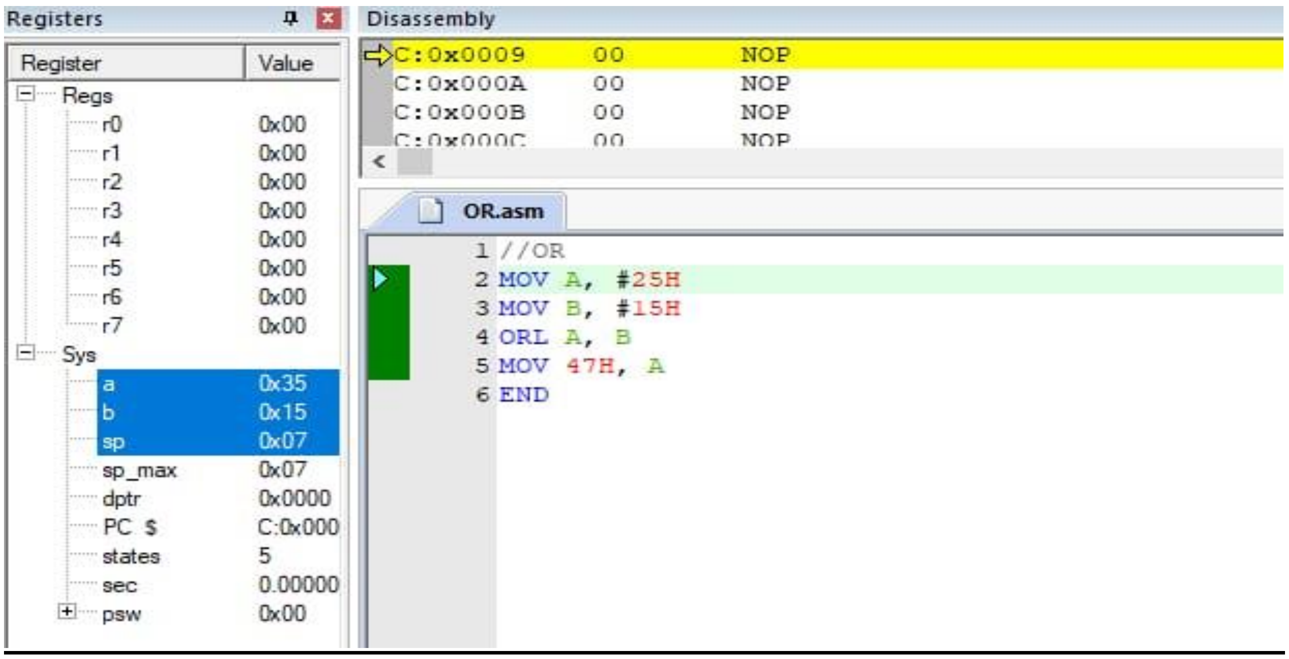
division.asm

```
ORG 000H    ; Define the origin of the program
MAIN:
    MOV A, #48H
    MOV B, #4H
    DIV AB
    MOV R0, A
END
```

OR PROGRAM:

```
ORG 000H   ; Define the origin of the program

MAIN:

MOV A, #25H

MOV B, #15H

ORL A, B

MOV 47H, A

END
```

OUTPUT



```
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:0x000
  states        5
  sec           0.00000
  psw           0x00
```

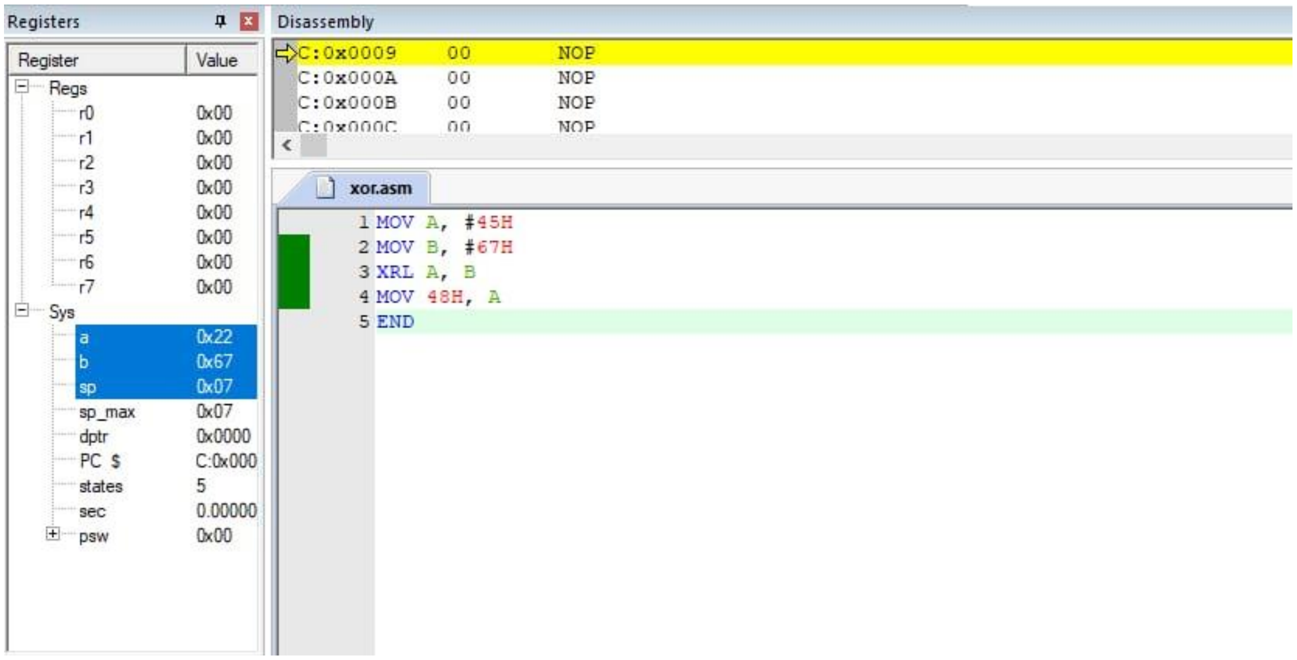
```
C:0x0009    00    NOP
C:0x000A    00    NOP
C:0x000B    00    NOP
C:0x000C    00    NOP
```

```
//OR
MOV A, #25H
MOV B, #15H
ORL A, B
MOV 47H, A
END
```

XOR PROGRAM:

```
ORG 000H   ; Define the origin of the program

MAIN:

MOV A, #45H

MOV B, #67H

XRL A, B

MOV 48H, A

END
```

OUTPUT:



```
Registers
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:0x000
  states    5
  sec       0.00000
  psw       0x00
```

```
Disassembly
C:0x0009      00        NOP
C:0x000A      00        NOP
C:0x000B      00        NOP
C:0x000C      00        NOP
```

```
xor.asm
MOV A, #45H
MOV B, #67H
XRL A, B
MOV 48H, A
END
```

RESULT:

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

EXP NO. : 04	**BASIC ARITHMETIC PROGRAM USING EMBEDDED C**
DATE:	

AIM:

To write a basic arithmetic Program using Embedded C

APPARATUS REQUIRED

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

PROCEDURE:

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

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

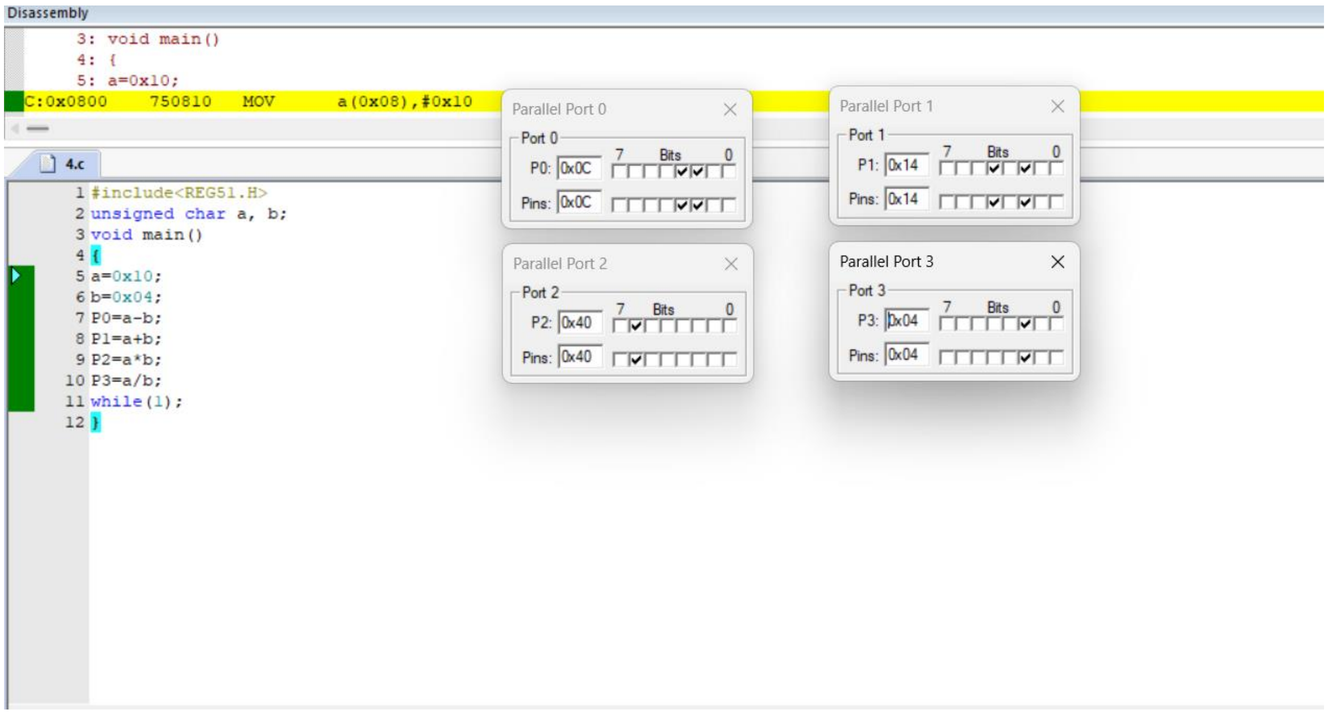
Step 3 : Debug the Code using Debug → Start / Stop Debug Session.

Step 4 : Stop the Program.

PROGRAM:

```
#include<REG51.H>
unsigned char a, b;
void main()
{
a=0x10;
b=0x04;
P0=a-b;
P1=a+b;
P2=a*b;
P3=a/b;
while(1);
}
```

OUTPUT:



```
     3: void main()
     4: {
     5: a=0x10;
C:0x0800    750810   MOV      a(0x08),#0x10
```

```
#include<REG51.H>
unsigned char a, b;
void main()
{
a=0x10;
b=0x04;
P0=a-b;
P1=a+b;
P2=a*b;
P3=a/b;
while(1);
}
```

Parallel Port 0 — Port 0 — P0: 0x0C — Pins: 0x0C

Parallel Port 1 — Port 1 — P1: 0x14 — Pins: 0x14

Parallel Port 2 — Port 2 — P2: 0x40 — Pins: 0x40

Parallel Port 3 — Port 3 — P3: 0x04 — Pins: 0x04

RESULT:

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

EXP NO.: 05	INTRODUCTION TO ARDUINO PLATFORM AND PROGRAMMING
DATE:	

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. Some components may differ depending on the type of board. 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. Let's discuss some common and best Arduino boards.

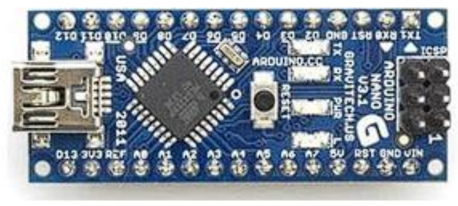
TYPES OF ARDUINO BOARDS

1) Arduino UNO

Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The Arduino UNO includes 6 analogue pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is the most used and of standard form from the list of all available Arduino Boards.

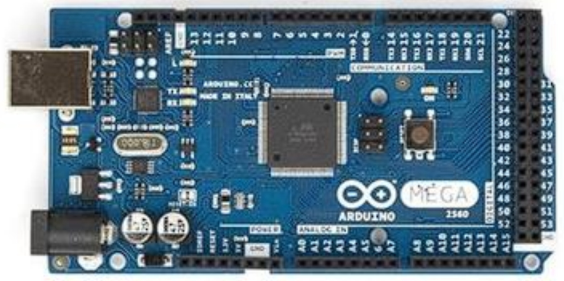
2) Arduino Nano

The Arduino Nano is a small Arduino board based on ATmega328P or ATmega628 Microcontroller. The connectivity is the same as the Arduino UNO board. The Nano board is defined as a sustainable, small, consistent, and flexible microcontroller board. It is small in size compared to the UNO board. The devices required to start our projects using the Arduino Nano board are Arduino IDE and miniUSB. The Arduino Nano includes an I/O pin set of 14 digital pins and 8 analogue pins. It also includes 6 Power pins and 2 Reset pins.



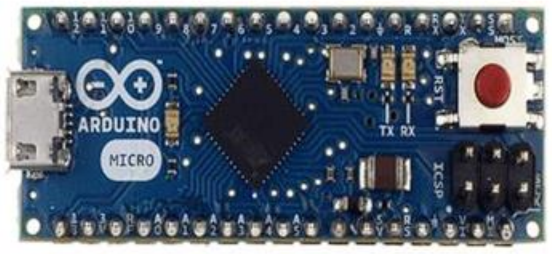
3) Arduino Mega

The Arduino Mega is based on the ATmega2560 Microcontroller. The ATmega2560 is an 8-bit microcontroller. We need a simple USB cable to connect to the computer and the AC to DC adapter or battery to get started with it. It has the advantage of working with more memory space. The Arduino Mega includes 54 I/O digital pins and 16 Analogue Input/Output (I/O), ICSP header, a reset button, 4 UART (Universal Asynchronous Receiver/Transmitter) ports, USB connection, and a power jack.



4) Arduino Micro

The Arduino Micro is based on the ATmega32U4 Microcontroller. It consists of 20 sets of pins. The 7 pins from the set are PWM (Pulse Width Modulation) pins, while 12 pins are analogue input pins. The other components on board are a reset button, a 16MHz crystal oscillator, an ICSP header, and a micro-USB connection. The USB is built in the Arduino Micro board.



5) Arduino Leonardo

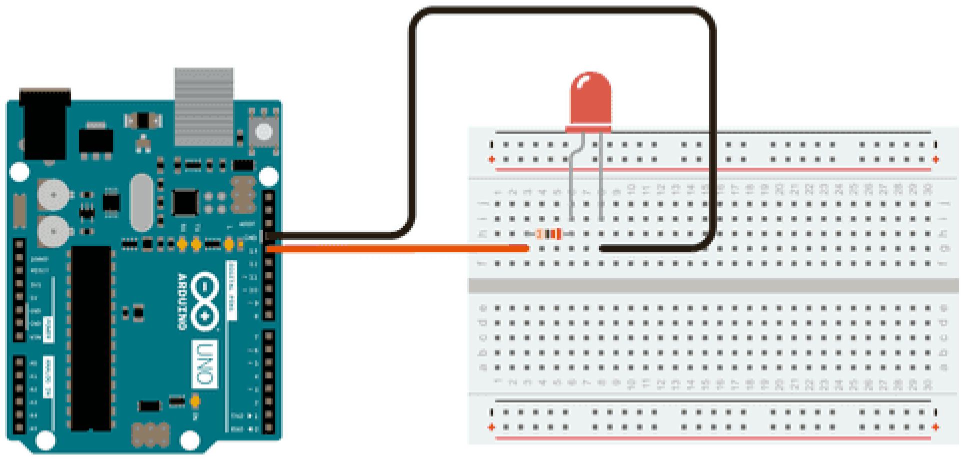
The basic specification of the Arduino Leonardo is the same as the Arduino Micro. It is also based on the ATmega32U4 Microcontroller. The components present on the board are 20 analogue and digital pins, a reset button, a 16MHz crystal oscillator, an ICSP header, and a micro USB connection.

PROGRAM:

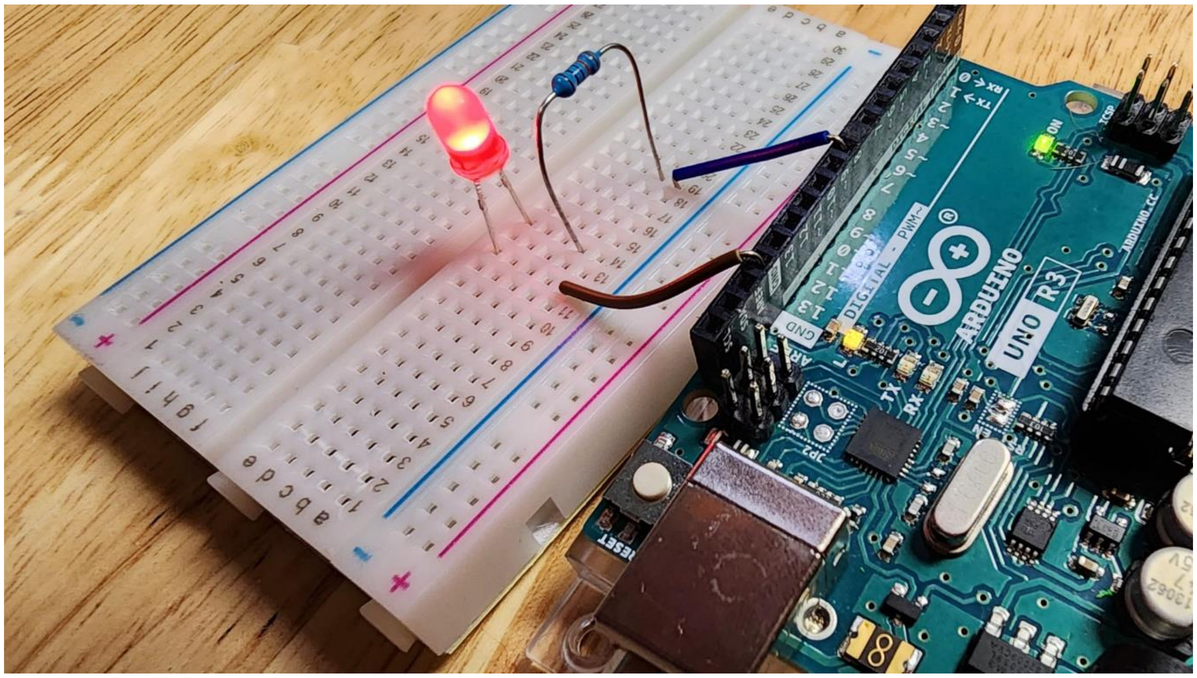
DIGITAL WRITE:

```
void setup() {
pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```

CIRCUIT DIAGRAM



OUTPUT:



RESULT:

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

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

AIM:

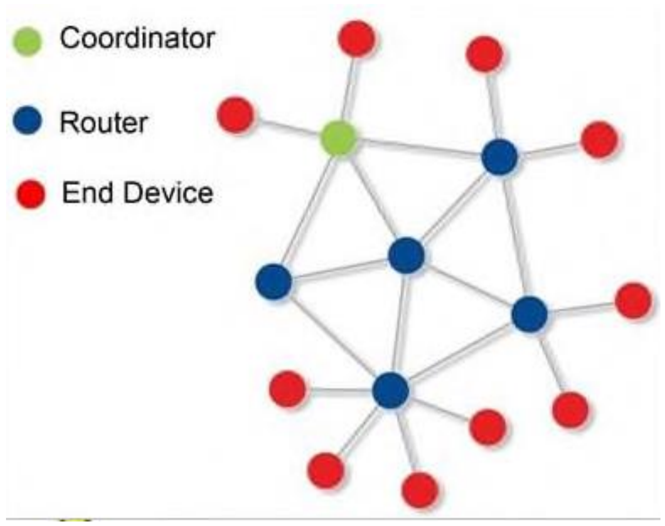
To Explore different communication methods with IoT devices (Zigbee, GSM, Bluetooth).

DIFFERENT COMMUNICATION METHODS:

IoT devices require reliable and efficient communication methods to transmit data and interact with other devices or systems. Here are three commonly used communication methods for IoT devices:

Zigbee:

Zigbee is a low-power wireless communication protocol designed for short-range communication between devices. It operates on the 2.4 GHz frequency band and supports mesh networking, allowing devices to communicate with each other through intermediate nodes. Zigbee is commonly used in home automation, industrial control, and smart energy applications.



GSM (Global System for Mobile Communications):

GSM is a widely used cellular network technology that enables IoT devices to connect to the internet using SIM cards. It operates on various frequency bands and provides wide coverage, making it suitable for applications that require long-range communication. GSM is commonly used in applications such as asset tracking, remote monitoring, and smart cities.

PROGRAM TO CONTROL AN LED USING A BLUETOOTH MODULE

AIM:

To write a program to control an LED using a Bluetooth module.

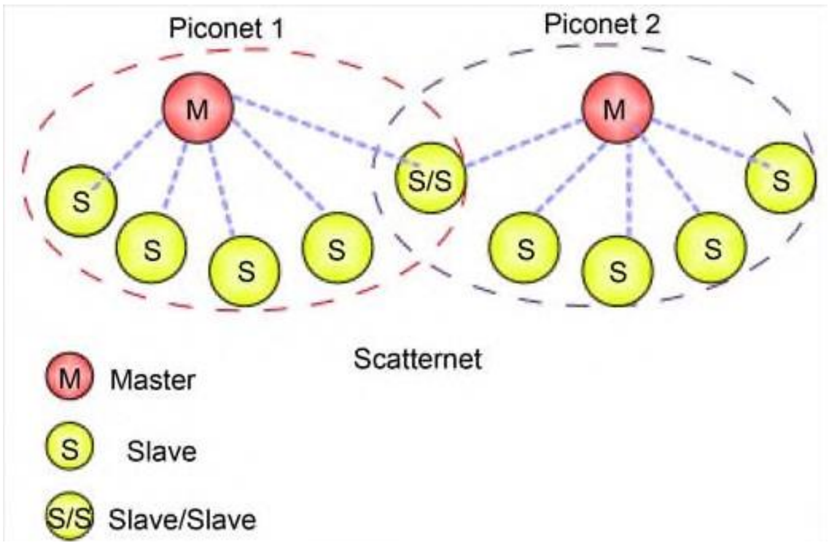
HARDWARE & SOFTWARE TOOLS REQUIRED:

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 short-range wireless communication technology that operates on the 2.4 GHz frequency band. It is commonly used for connecting IoT devices to smartphones, tablets, and other nearby devices. Bluetooth Low Energy (BLE) is a power-efficient version of Bluetooth that is ideal for battery-powered IoT devices. Bluetooth is widely used in applications such as wearable devices, healthcare monitoring, and home automation. Each communication method has its advantages and limitations, and the choice depends on the specific requirements of the IoT application.



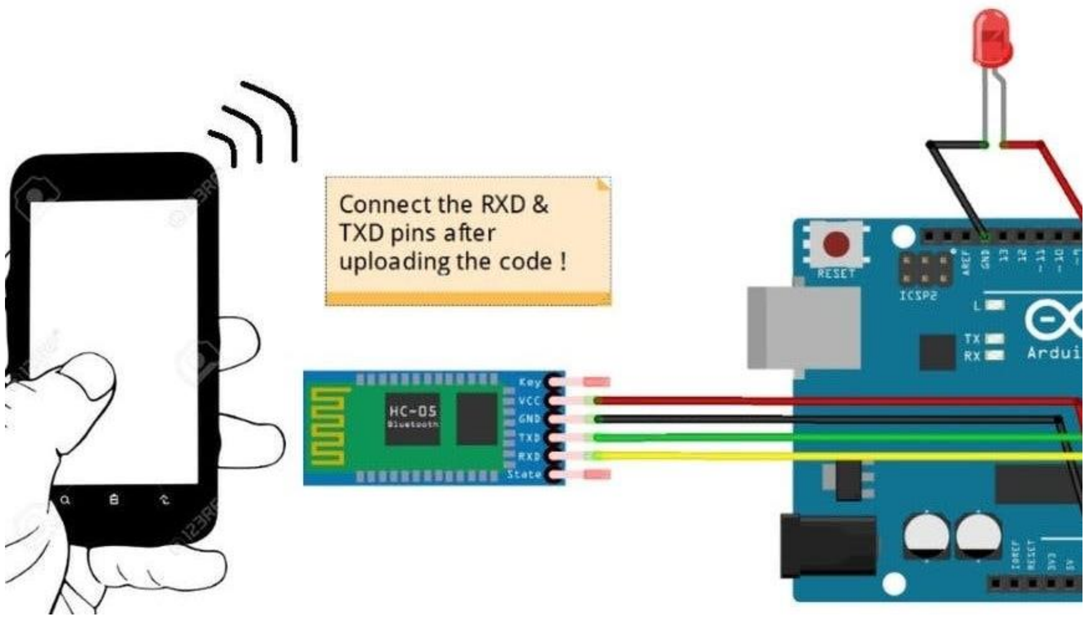
CONNECTIONS:

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

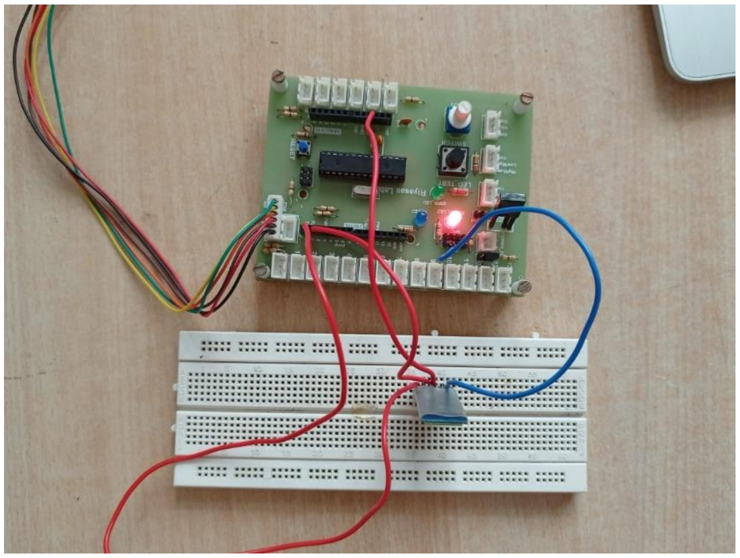
PROGRAM:

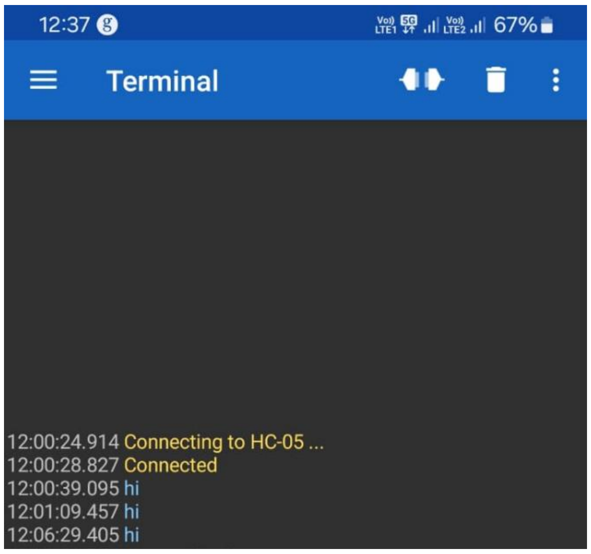
```
#include<SoftwareSerial.h>
SoftwareSerial mySerial(2,3); //rx,tx
void setup() {
mySerial.begin(9600);
Serial.begin(9600);
pinMode(4, OUTPUT);
}
void loop() {
if(mySerial.available()>0)
{
char data=mySerial.read();
Serial.println(data);
if(data=='1'){
digitalWrite(4,HIGH);
Serial.println("LED ON");
}
else if(data=='2'){
digitalWrite(4,LOW);
Serial.println("LED OFF");
}
}
}
```

CIRCUIT DIAGRAM:



OUTPUT:





```
12:00:24.914 Connecting to HC-05 ...
12:00:28.827 Connected
12:00:39.095 hi
12:01:09.457 hi
12:06:29.405 hi
```

RESULT:

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

EXP NO. : 07	INTRODUCTION TO RASPBERRY PI PLATFORM AND PYTHON PROGRAMMING
DATE:	

AIM:

To study the Raspberry Pi platform and python programming.

Introduction to Raspberry Pi Pico W:

The Raspberry Pi Pico W is a compact and affordable microcontroller board developed by the Raspberry Pi Foundation. Building upon the success of the Raspberry Pi Pico, the Pico W variant brings wireless connectivity to the table, making it an even more versatile platform for embedded projects. In this article, we will provide a comprehensive overview of the Raspberry Pi Pico W, highlighting its key features and capabilities.

Features:

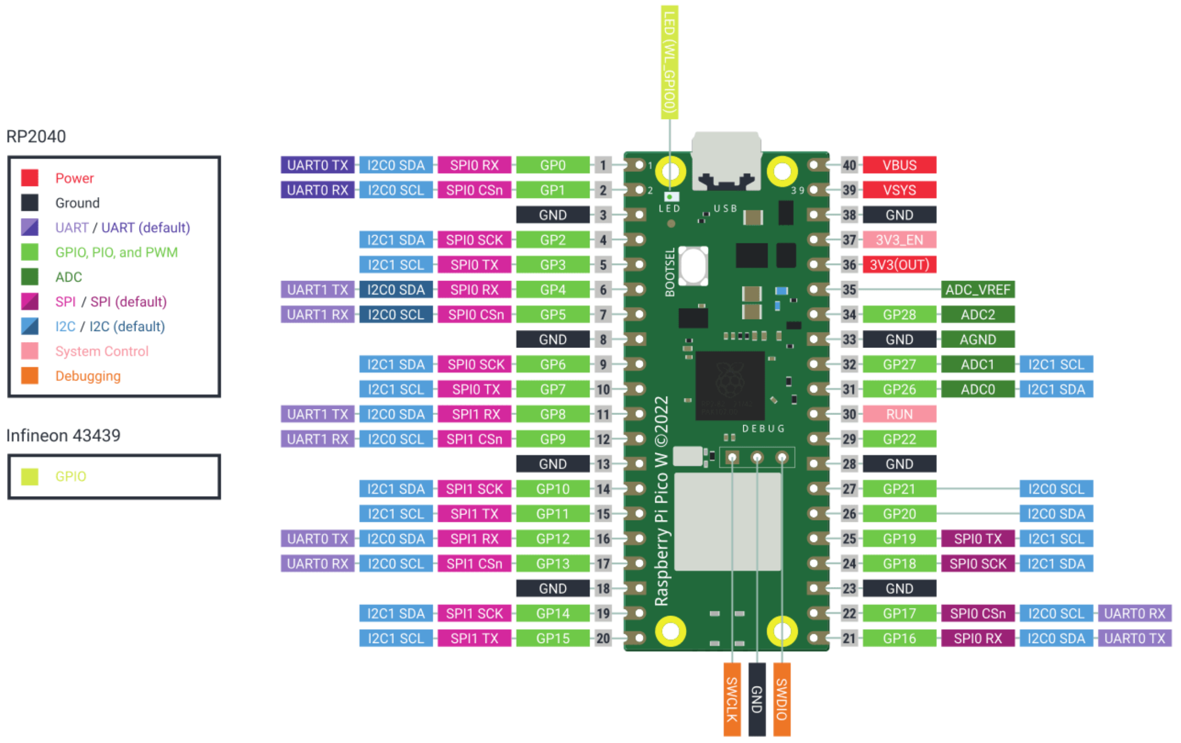
- RP2040 microcontroller with 2MB of flash memory
- On-board single-band 2.4GHz wireless interfaces (802.11n)
- Micro USB B port for power and data (and for reprogramming the flash)
- 40 pins 21mmx51mm 'DIP' style 1mm thick PCB with 0.1″ through-hole pins also with edge
- castellations
- Exposes 26 multi-function 3.3V general purpose I/O (GPIO)
- 23 GPIO are digital-only, with three also being ADC-capable
- Can be surface mounted as a module
- 3-pin ARM serial wire debug (SWD) port
- Simple yet highly flexible power supply architecture
- Various options for easily powering the unit from micro-USB, external supplies, or batteries
- High quality, low cost, high availability
- Comprehensive SDK, software examples, and documentation
- Dual-core Cortex M0+ at up to 133MHz
- On-chip PLL allows variable core frequency
- 264kByte multi-bank high-performance SRAM

MicroPython and C/C++ Programming:

The Raspberry Pi Pico W can be programmed using MicroPython, a beginner-friendly programming Language that allows for rapid prototyping and development. MicroPython provides a simplified syntax and high-level abstractions, making it easy for newcomers to get started. Additionally, the Pico W is compatible with C/C++ programming, allowing experienced developers to leverage the rich ecosystem of libraries and frameworks available.

Programmable Input/Output (PIO) State Machines:

One of the unique features of the RP2040 microcontroller is the inclusion of Programmable Input/Output (PIO) state machines. These state machines provide additional processing power and flexibility for handling real-time data and timing-critical applications. The PIO state machines can be programmed to interface with custom protocols, generate precise waveforms, and offload tasks from the main processor, enhancing the overall performance of the system. Open-Source and Community Support. As with all Raspberry Pi products, the Pico W benefits from the vibrant and supportive Raspberry Pi community. Raspberry Pi provides extensive documentation, including datasheets, pinout diagrams, and programming guides, to assist developers in understanding the board's capabilities. The community offers forums, online tutorials, and project repositories, allowing users to seek help, share knowledge, and collaborate on innovative projects.

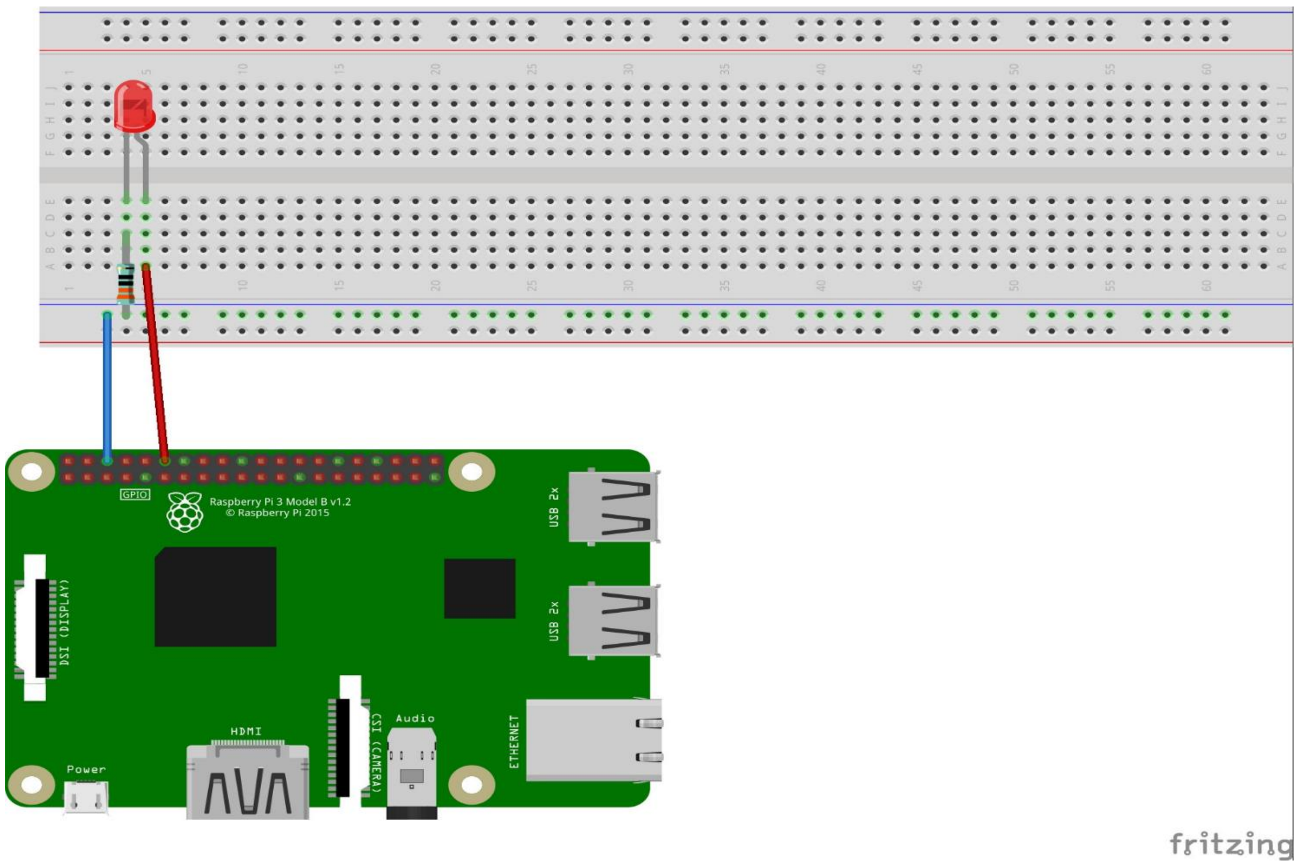


PROGRAM

LED:

```
from machine import Pin
import time
LED = Pin(16, Pin.OUT)
while True:
LED.value(1)
time.sleep(1)
LED.value(0)
time.sleep(1)
```

OUTPUT



RESULT:

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

EXP NO. : 08	INTERFACING SENSORS WITH RASPBERRY PI
DATE:	

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

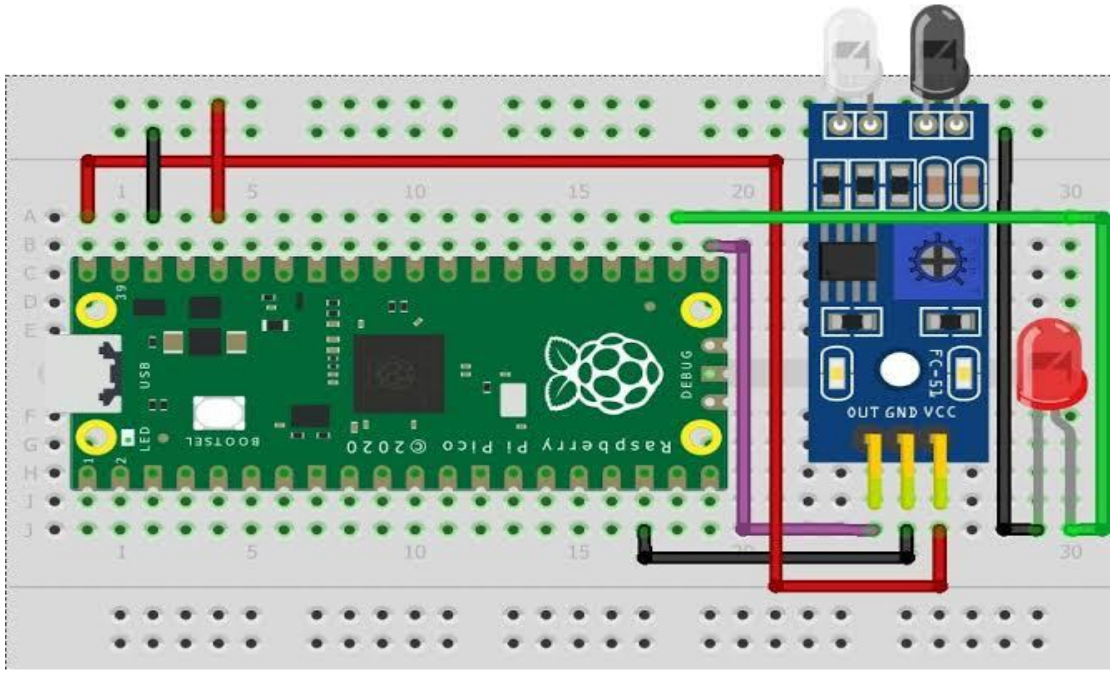
S.NO	Hardware and software requirement	Quantity
1	Thonny IDE	1
2	Raspberry Pi pico Development Board	1
3	Jump wires	Few
4	Micro USB Cable	1
5	IR Sensor & Ultrasonic Sensor	1

PROGRAM:

IR SENSOR:

```
from machine import Pin
from time import sleep
buzzer=Pin(16,Pin.OUT)
ir=Pin(15,Pin.IN)
while True:
ir_value=ir.value()
if ir_value== True:
print("Buzzer OFF")
buzzer.value(0)
else:
print("Buzzer ON")
buzzer.value (1)
sleep(0.5)
```

OUTPUT:

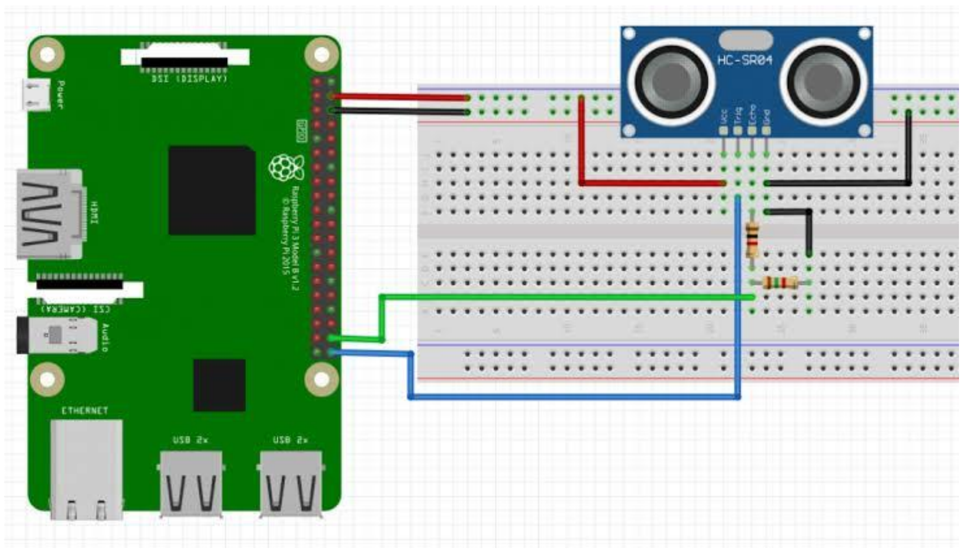


ULTRASONIC SENSOR:

```
from machine import Pin, PWM
import utime
trigger = Pin(14, Pin.OUT)
echo = Pin(15, Pin.IN)
buzzer = Pin(16, Pin.OUT)
def measure_distance():
trigger.low()
utime.sleep_us(2)
trigger.high()
utime.sleep_us(5)
trigger.low()
while echo.value() == 0:
signaloff = utime.ticks_us()
while echo.value() == 1:
signalon = utime.ticks_us()
timepassed = signalon - signaloff
distance = (timepassed * 0.0343) / 2
return distance
```

```
while True:
dist = measure_distance()
print(f"Distance : {dist} cm")
if dist <= 10:
buzzer.value(1)
utime.sleep(0.01)
else:
buzzer.value(0)
utime.sleep(0.01)
utime.sleep(0.5)
```

OUTPUT:



RESULT:

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

EXP NO. : 09	**COMMUNICATION BETWEEN ARDUINO AND RASPBERRY PI USING ANY WIRELESS MEDIUM**
DATE:	

AIM:

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

APPARATUS REQUIRED

S.NO	Hardware and software requirement	Quantity
1	Thonny IDE	1
2	Raspberry Pi pico Development Board	1
3	Arduino UNO Board	
3	Jump wires	Few
4	Micro USB Cable	1

PROGRAM:

MASTER- ARDUINO:

```
#include<SoftwareSerial.h>
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
2	-	Tx
3	-	Rx
-	GND	GND
-	5V	5V

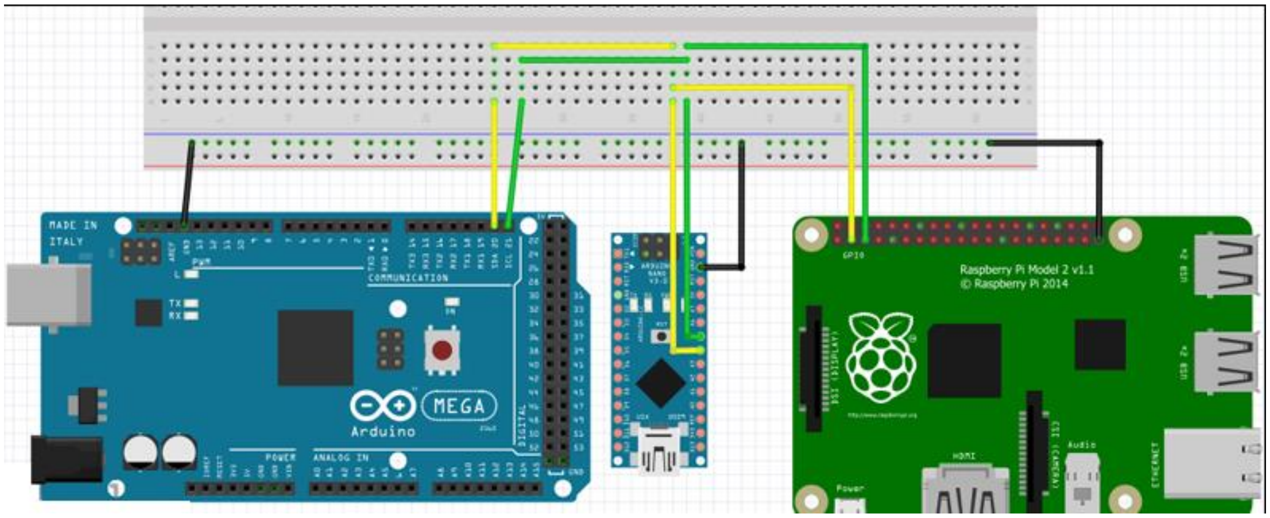
SLAVE - RASPBERRY PI PICO

```
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')
```

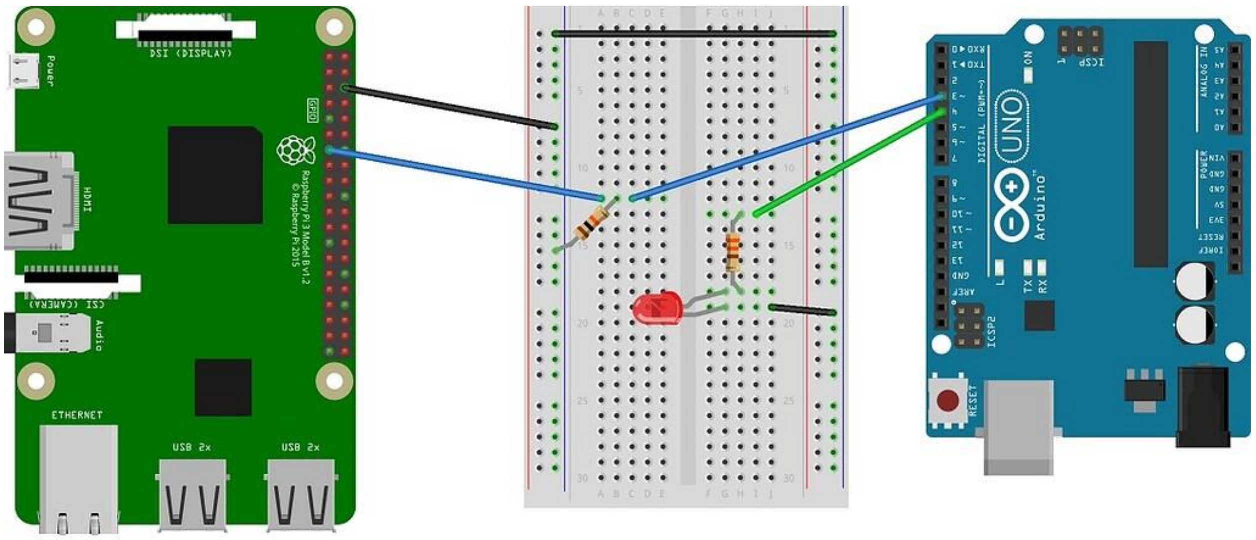
CONNECTIONS:

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

OUTPUT:



<u>INTERFACING FOR LED</u>



RESULT:

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

EXP NO. : 10	SETUP A CLOUD PLATFORM TO LOG THE DATA
DATE:	

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.NO.	SOFTWARE REQUIREMENT	QUANTITY
1	Blynk Platform	1

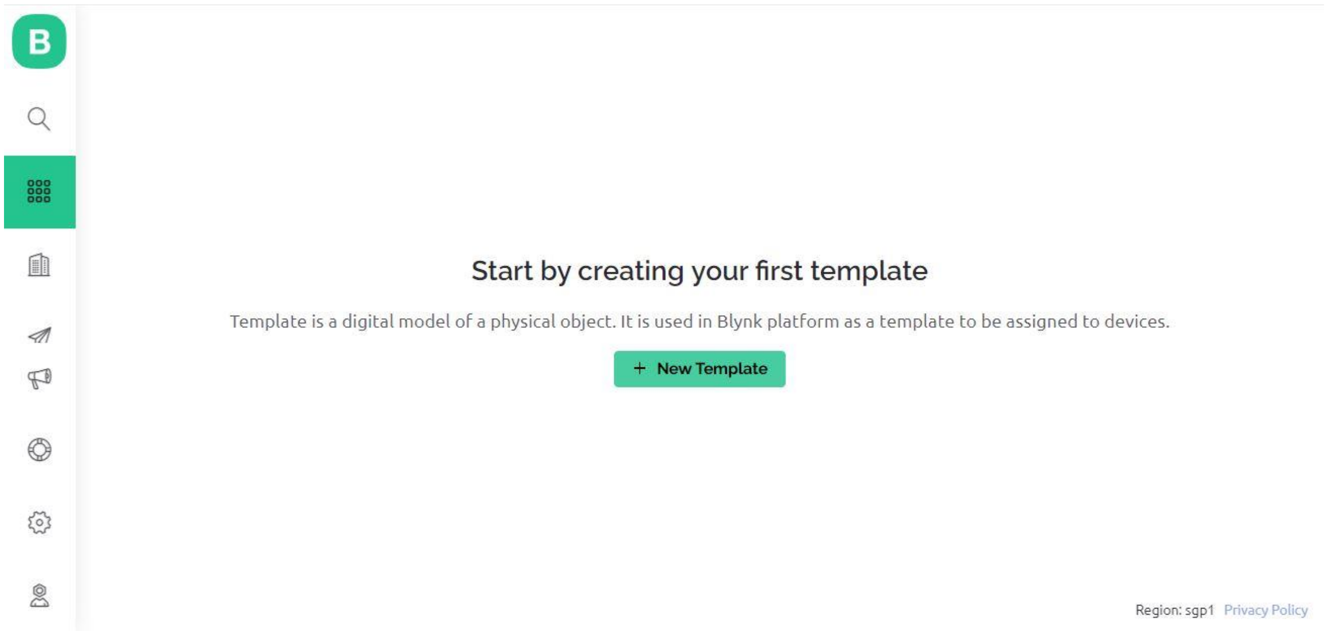
CLOUD PLATFORM-BLYNK:

Blynk is a smart platform that allows users to create their Internet of Things applications without the need for coding or electronics knowledge. It is based on the idea of physical programming & provides a platform to create and control devices where users can connect physical devices to the Internet and control them using a mobile app.

Setting up Blynk 2.0 Application .To control the LED using Blynk and Raspberry Pi Pico W, you need to create a Blynk project and set up a dashboard in the mobile or web application. Here's how you can set up the dashboard:

Step 1: Visit blynk.cloud and create a Blynk account on the Blynk website. Or you can simply sign in using the registered Email ID.

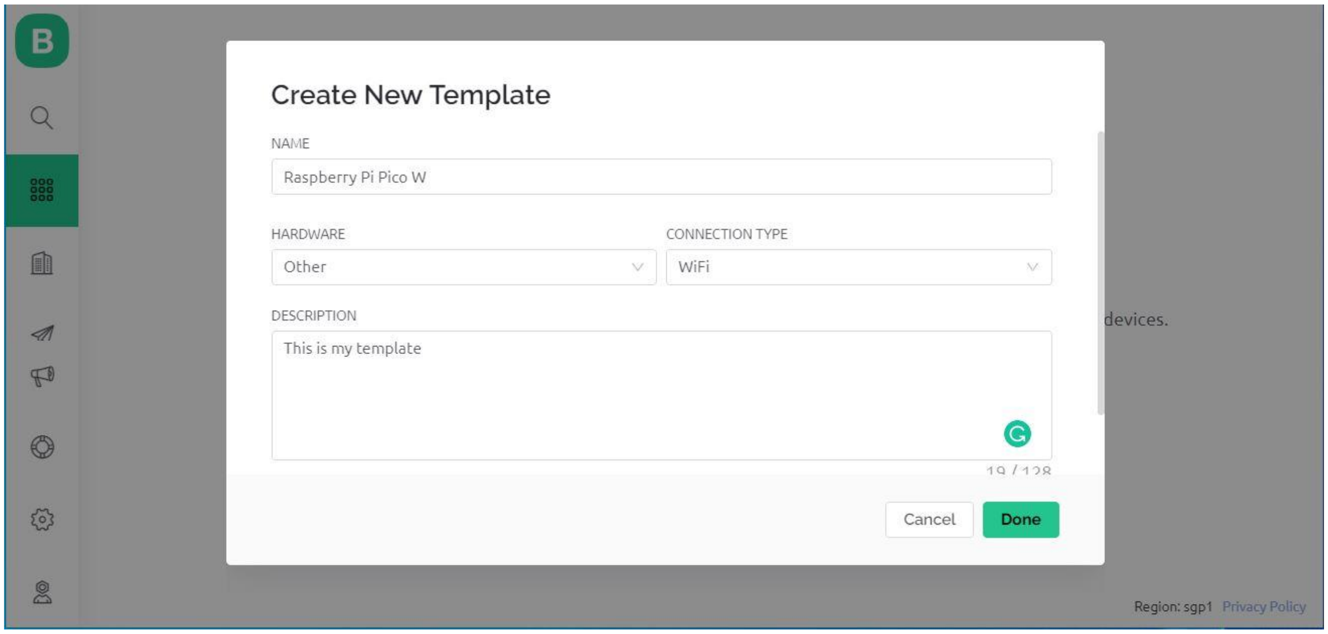
Step 2: Click on +New Template.



Start by creating your first template

Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.

Step 3: Give any name to the Template such as Raspberry Pi Pico W. Select ‘Hardware Type’ as Other and ‘Connection Type’ as WiFi.



Create New Template

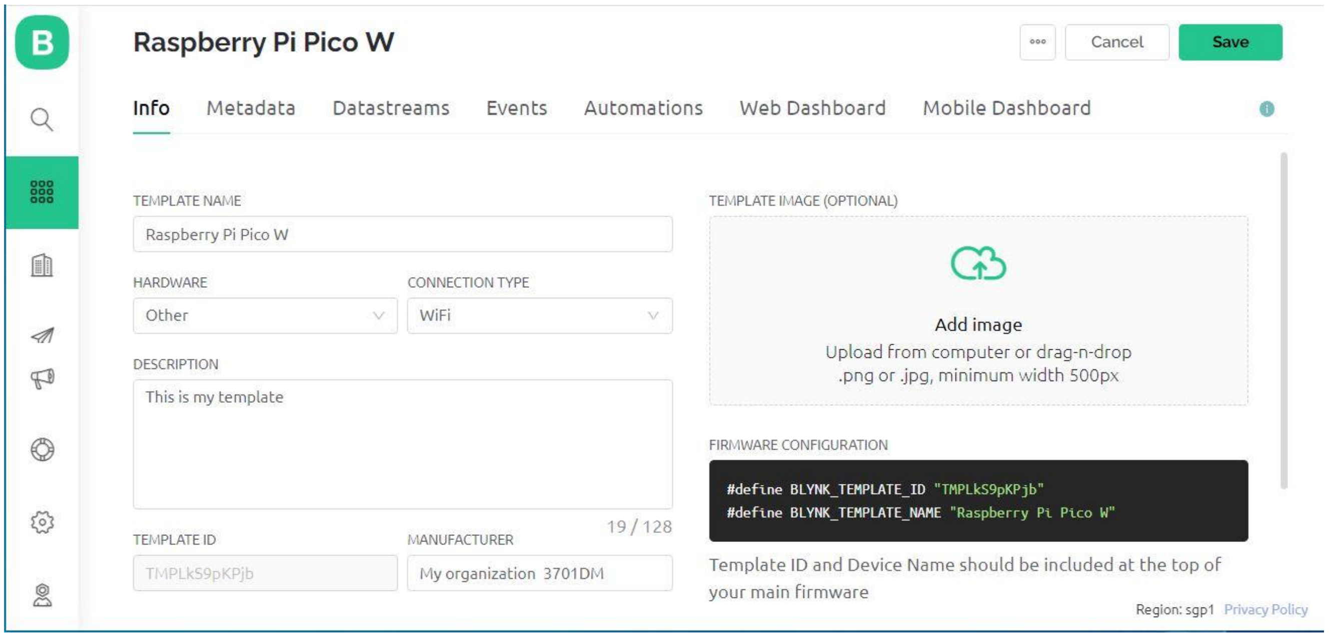
NAME: Raspberry Pi Pico W

HARDWARE: Other

CONNECTION TYPE: WiFi

DESCRIPTION: This is my template

So a template will be created now.


Raspberry Pi Pico W

TEMPLATE NAME: Raspberry Pi Pico W

HARDWARE: Other

CONNECTION TYPE: WiFi

DESCRIPTION: This is my template

TEMPLATE ID: TMPLkS9pKPjb

MANUFACTURER: My organization 3701DM

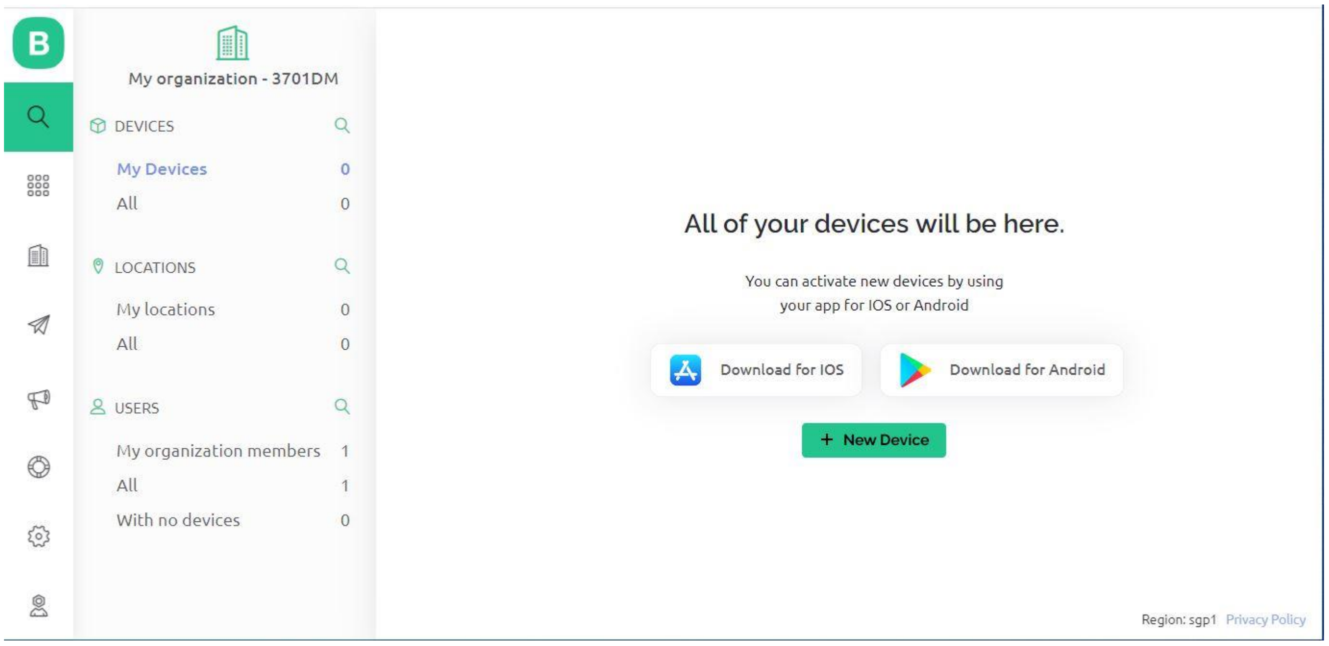
FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPLkS9pKPjb"
#define BLYNK_TEMPLATE_NAME "Raspberry Pi Pico W"
```

Template ID and Device Name should be included at the top of your main firmware

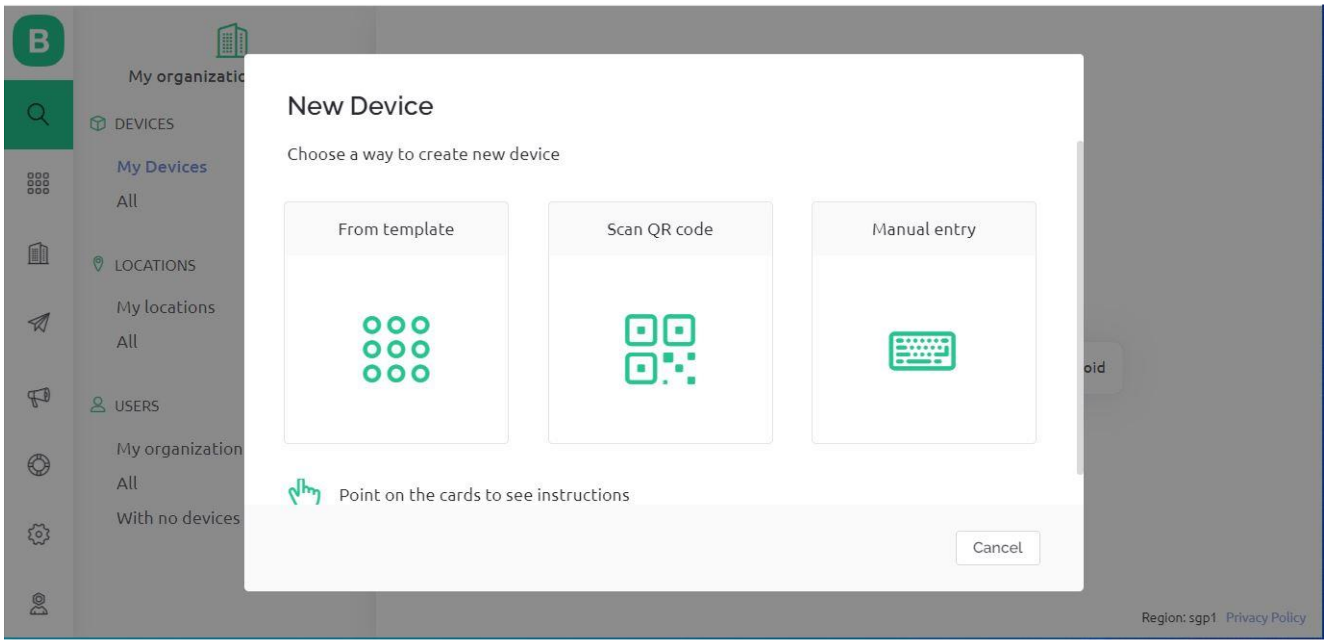
Cancel
Save
Info
Metadata
Datastreams
Events
Automations
Web Dashboard
Mobile Dashboard
TEMPLATE IMAGE (OPTIONAL)
Add image
Upload from computer or drag-n-drop
.png or .jpg, minimum width 500px
19 / 128
Region: sgp1 Privacy Policy

Step 4: Now we need to add a 'New Device' now.

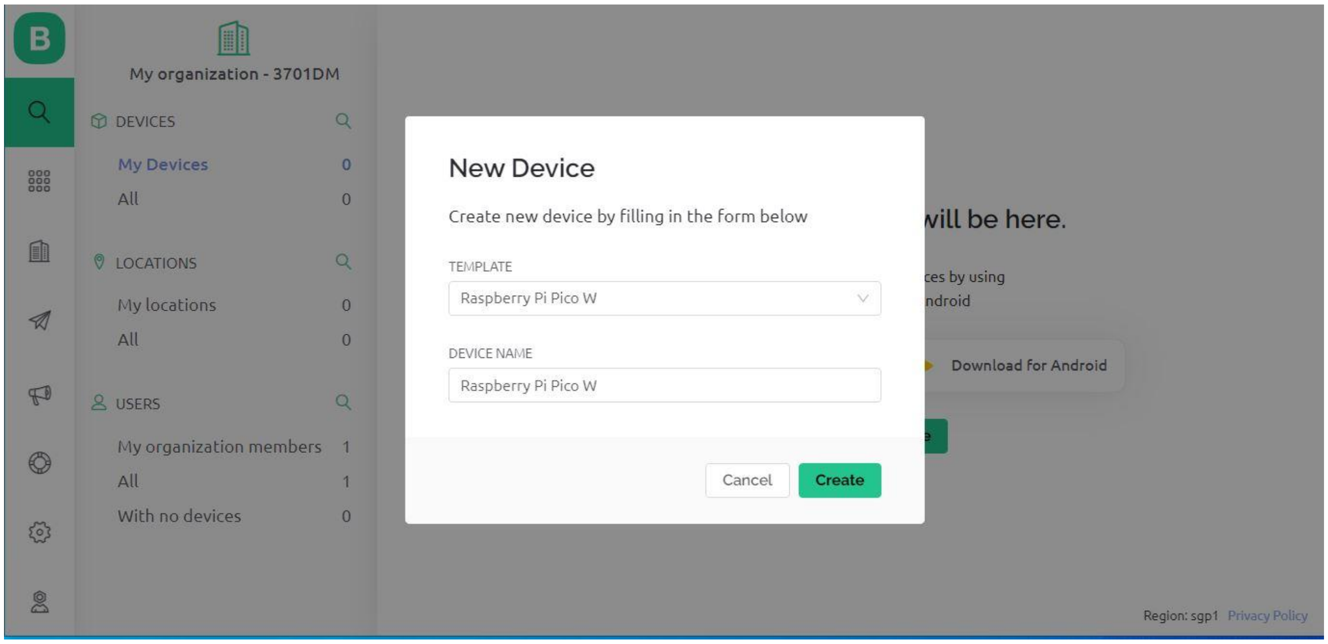

All of your devices will be here.

You can activate new devices by using your app for IOS or Android

Select a New Device from 'Template'.



Select the device from a template that you created earlier and also give any name to the device. Click on Create.



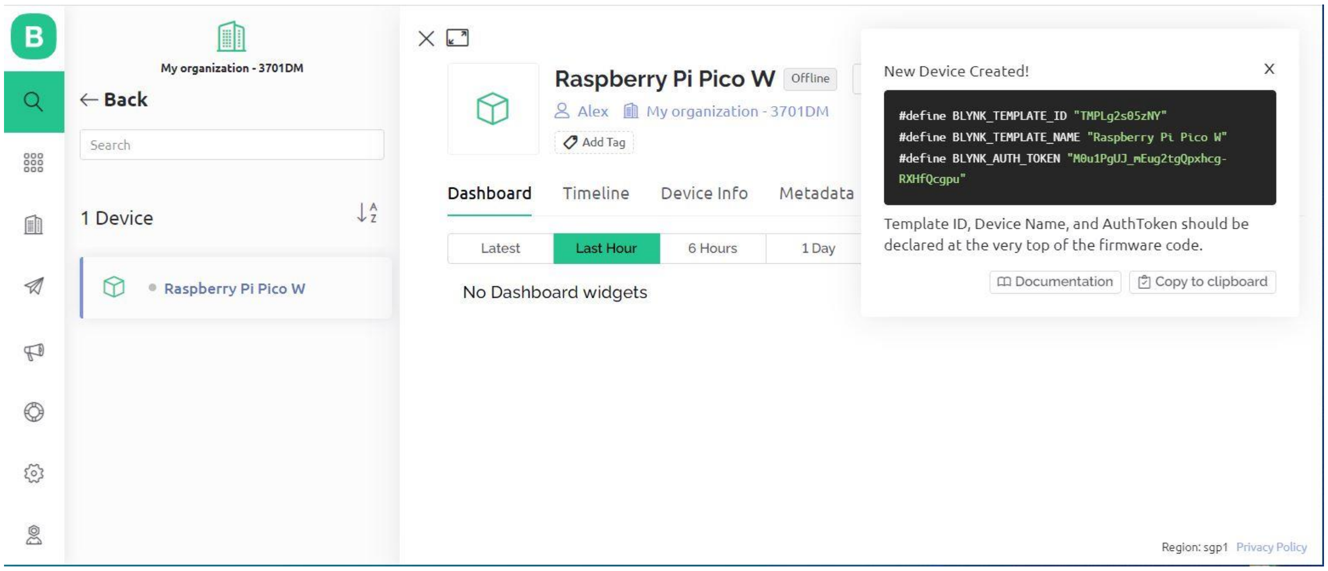
New Device

Create new device by filling in the form below

TEMPLATE: Raspberry Pi Pico W

DEVICE NAME: Raspberry Pi Pico W

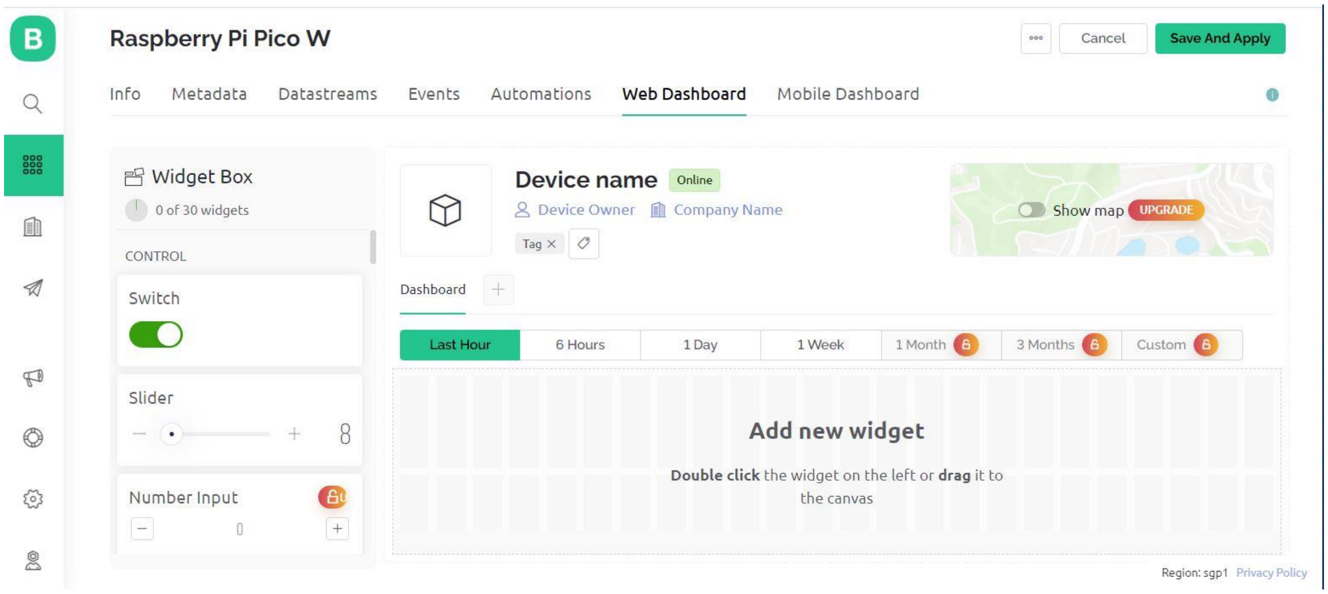
A new device will be created. You will find the Blynk Authentication Token Here. Copy it as it is necessary for the code.


New Device Created!

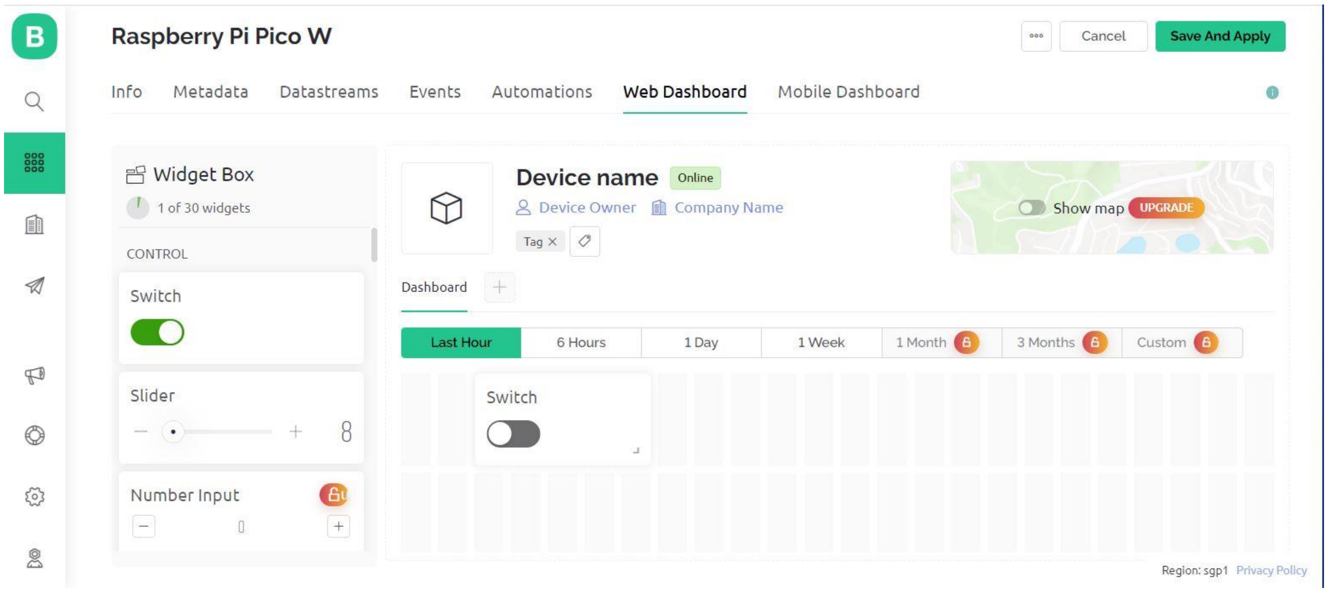
```
#define BLYNK_TEMPLATE_ID "TMPLg2s05zNY"
#define BLYNK_TEMPLATE_NAME "Raspberry Pi Pico W"
#define BLYNK_AUTH_TOKEN "M0u1PgUJ_mEug2tgQpxhcg-
RXHfQcgpu"
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.

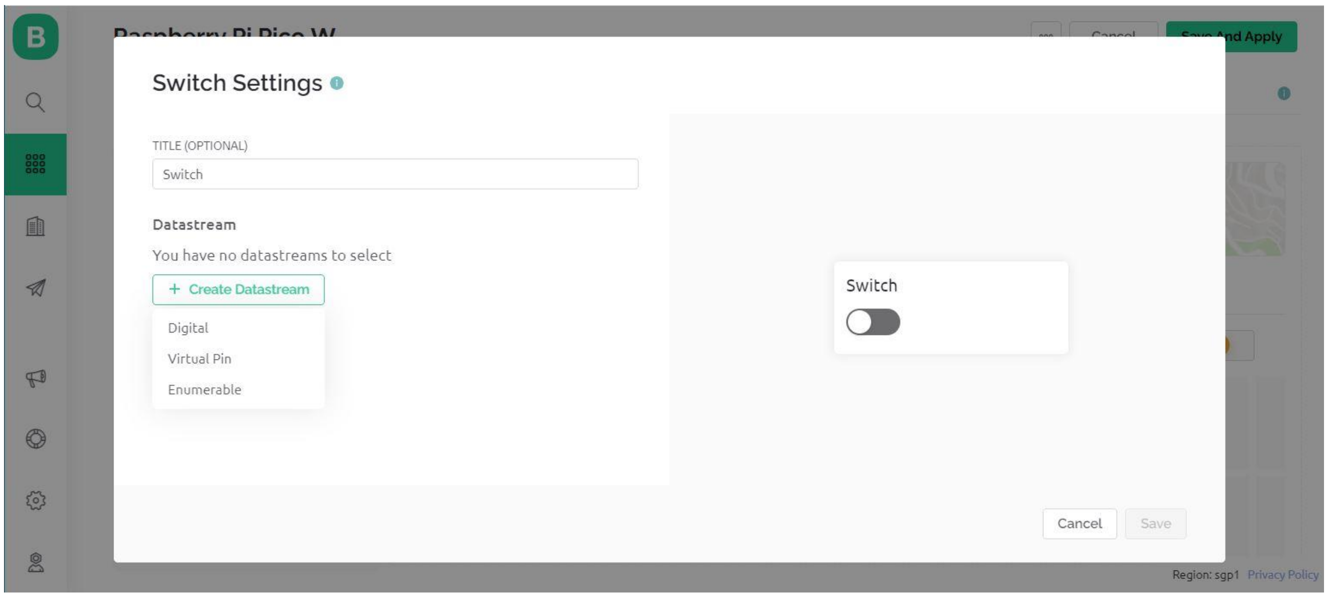
Step 5: Now go to the dashboard and select 'Web Dashboard'.



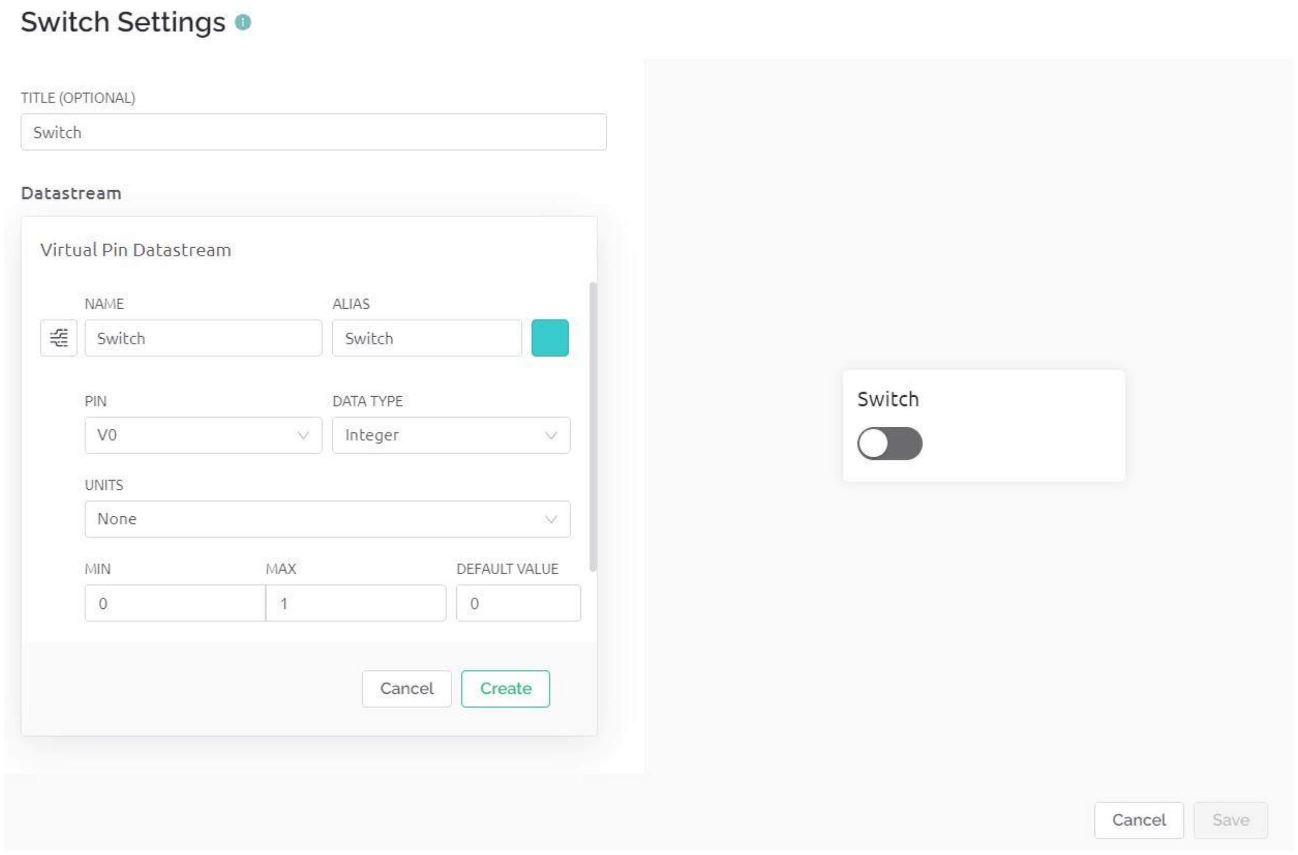
From the widget box drag a switch and place it on the dashboard screen.



Step 6: On the switch board click on Settings and here you need to set up the Switch. Give any title to it and Create DataStream as Virtual Pin.



Configure the switch settings as per the image below and click on create.



Switch Settings

TITLE (OPTIONAL): Switch

Datastream

Virtual Pin Datastream

NAME: Switch

ALIAS: Switch

PIN: V0

DATA TYPE: Integer

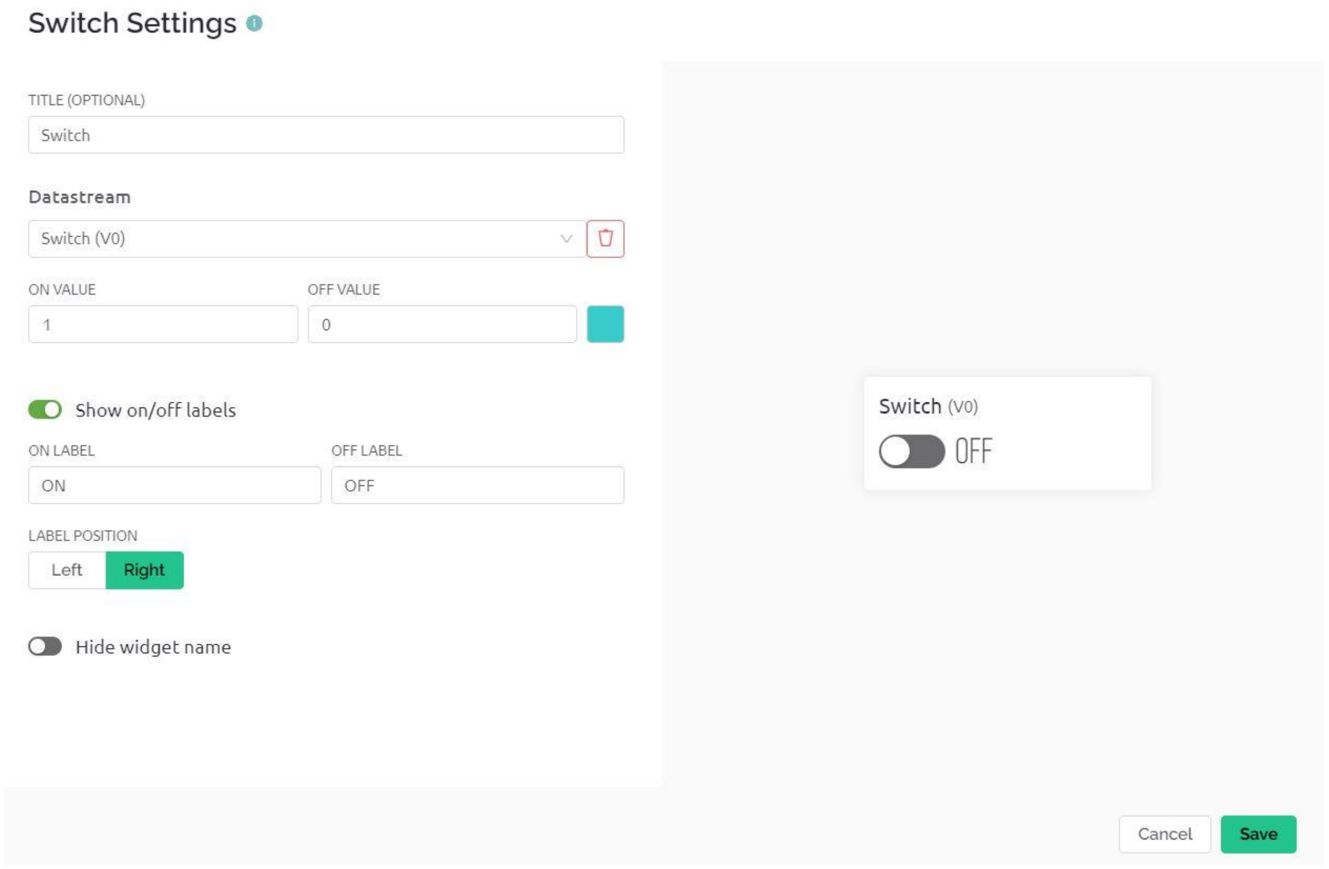
UNITS: None

MIN: 0

MAX: 1

DEFAULT VALUE: 0

Configure the final steps again.



Switch Settings

TITLE (OPTIONAL): Switch

Datastream: Switch (V0)

ON VALUE: 1

OFF VALUE: 0

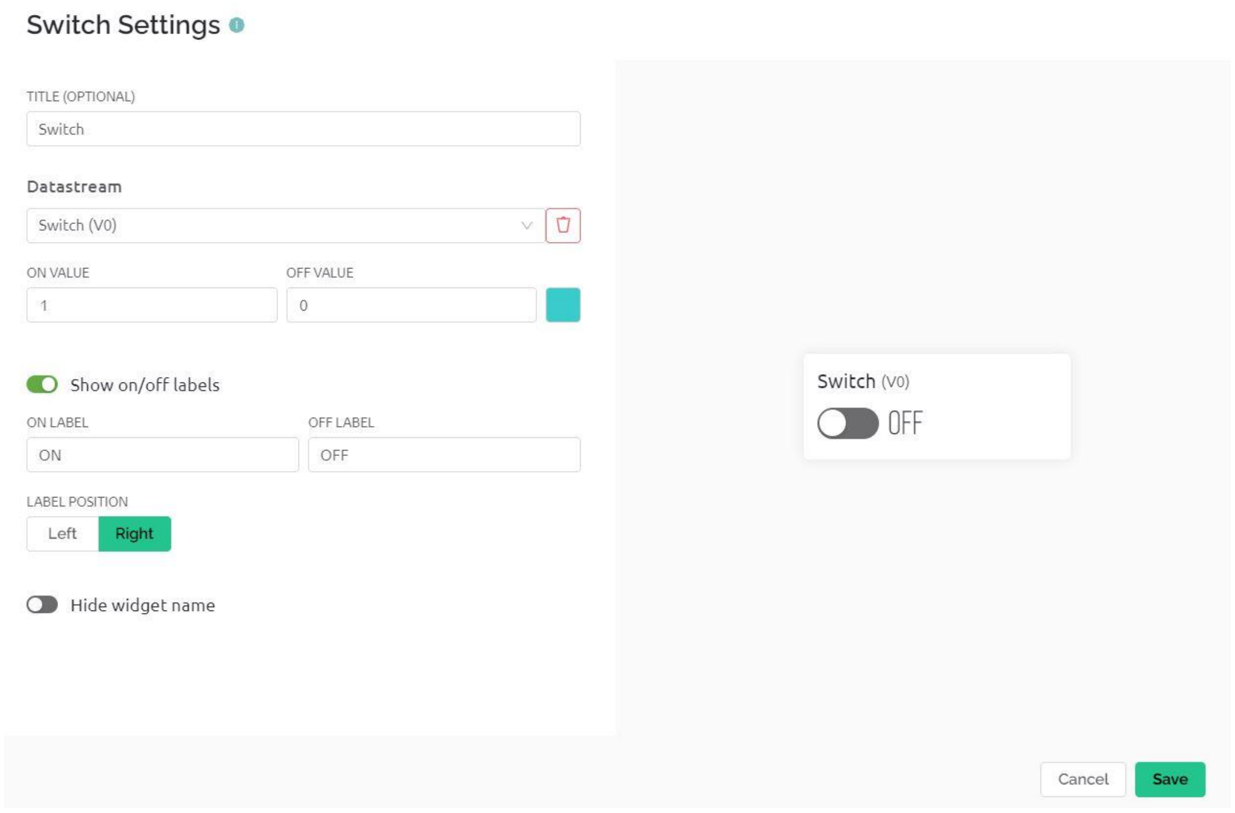
Show on/off labels

ON LABEL: ON

OFF LABEL: OFF

LABEL POSITION: Left [✓ Right]

Hide widget name



Switch Settings

TITLE (OPTIONAL)
Switch

Datastream
Switch (V0)

ON VALUE
1

OFF VALUE
0

Show on/off labels

ON LABEL
ON

OFF LABEL
OFF

LABEL POSITION
Left Right

Hide widget name

With this Blynk dashboard set up, you can now proceed to program the Raspberry Pi Pico W board to control the LED.

Step 7: To control the LED with a mobile App or Mobile Dashboard, you also need to setup the Mobile Phone Dashboard. The process is similarly explained above.

Install the Blynk app on your smartphone The Blynk app is available for iOS and Android. Download and install the app on your smartphone. then need to set up both the Mobile App and the Mobile Dashboard in order to control the LED with a mobile device. The process is explained above.

1. Open Google Play Store App on an android phone
2. Open Blynk.App
3. Log In to your account (using the same email and password)
4. Switch to Developer Mode
5. Find the "Raspberry Pi Pico Pico W" template we created on the web and tap on it
6. Tap on the "Raspberry Pi Pico Pico W" template (this template automatically comes because we created it on our dashboard).

7. tap on plus icon on the left-right side of the window

8. Add one button Switch

9. Now We Successfully Created an android template

10. it will work similarly to a web dashboard template

RESULT:

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

EXP NO.: 11	**LOG DATA USING RASPERRY PI AND UPLOAD TO THE CLOUN PLATFORM**
DATE:	

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

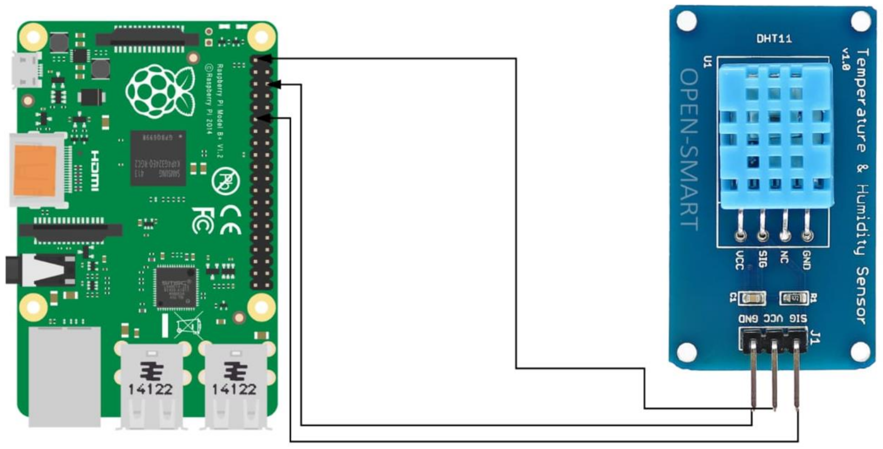
S.No.	Hardware & Software requirements	Quantity
1	Thonny IDE	1
2	Raspberry Pi Pico Development Board	Few
3	Jump Wires	1
4	Micro USB Cable	1

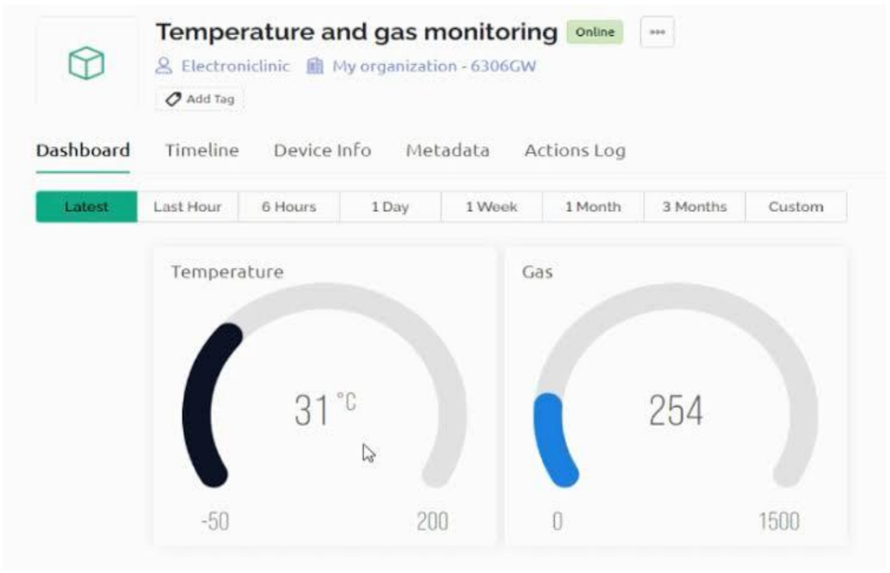
PROGRAM:

```
from machine import Pin, I2C, ADC
from utime import sleep_ms
from pico_i2c_lcd import I2cLcd
import time
import network
import BlynkLib
adc = machine.ADC(4)
i2c=I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
I2C_ADDR=i2c.scan()[0]
lcd=I2cLcd(i2c,I2C_ADDR,2,16)
wlan = network.WLAN()
wlan.active(True)
wlan.connect("Wifi_Username","Wifi_Password")
BLYNK_AUTH = 'Your_Token'
wait = 10
while wait > 0:
```

```
if wlan.status() < 0 or wlan.status() >= 3:
break
wait -= 1
print('waiting for connection...')
time.sleep(1)
if wlan.status() != 3:
raise RuntimeError('network connection failed')
else:
print('connected')
ip=wlan.ifconfig()[0]
print('IP: ', ip)
blynk = BlynkLib.Blynk(BLYNK_AUTH)
while True:
ADC_voltage = adc.read_u16() * (3.3 / (65536))
temperature_celcius = 27 - (ADC_voltage - 0.706)/0.001721
temp_fahrenheit=32+(1.8*temperature_celcius)
print("Temperature in C: {}".format(temperature_celcius))
print("Temperature in F: {}".format(temp_fahrenheit))
lcd.move_to(0,0)
lcd.putstr("Temp:")
lcd.putstr(str(round(temperature_celcius,2)))
lcd.putstr("C ")
lcd.move_to(0,1)
lcd.putstr("Temp:")
lcd.putstr(str(round(temp_fahrenheit,2)))
lcd.putstr("F")
time.sleep(5)
blynk.virtual_write(3, temperature_celcius)
blynk.virtual_write(4, temp_fahrenheit)
blynk.log_event(temperature_celcius)
blynk.run()
time.sleep(5)
```

OUTPUT:





RESULT:

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

EXP NO. **12**	**DESIGN AN IOT BASED SYSYTEM**
DATE:	

AIM:

To write a program using sensors for carparking assist.

APPARATUS REQUIRED :

S.NO	NAME OF THE EQUIPMENT / COMPONENT	QUANTITY
1	. PC with windows	1
2.	Arduino 1.6.5	1
3.	Proteus 8.0 with Senor Library	1

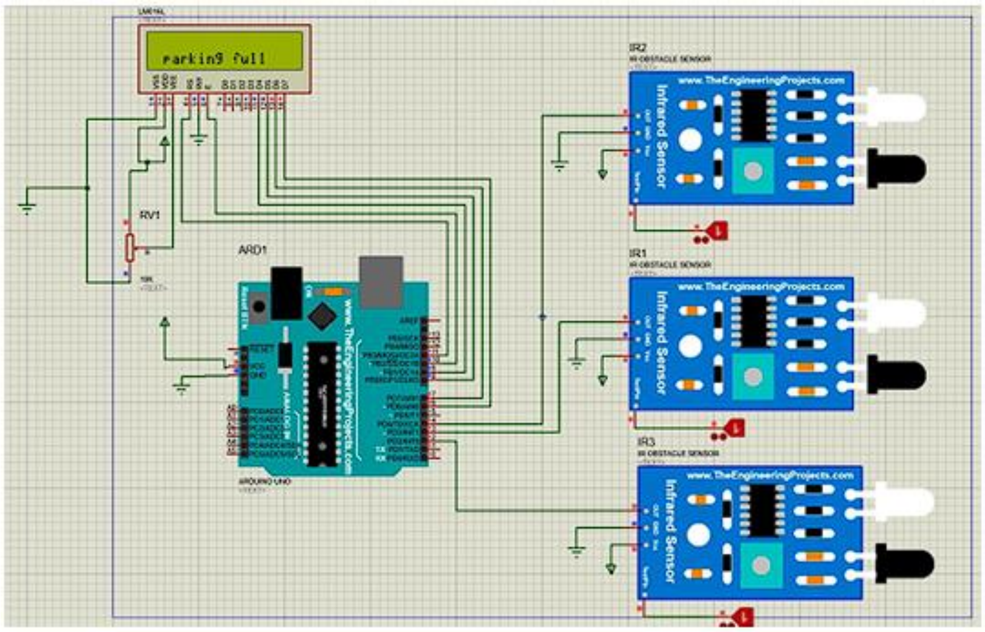
THEORY :

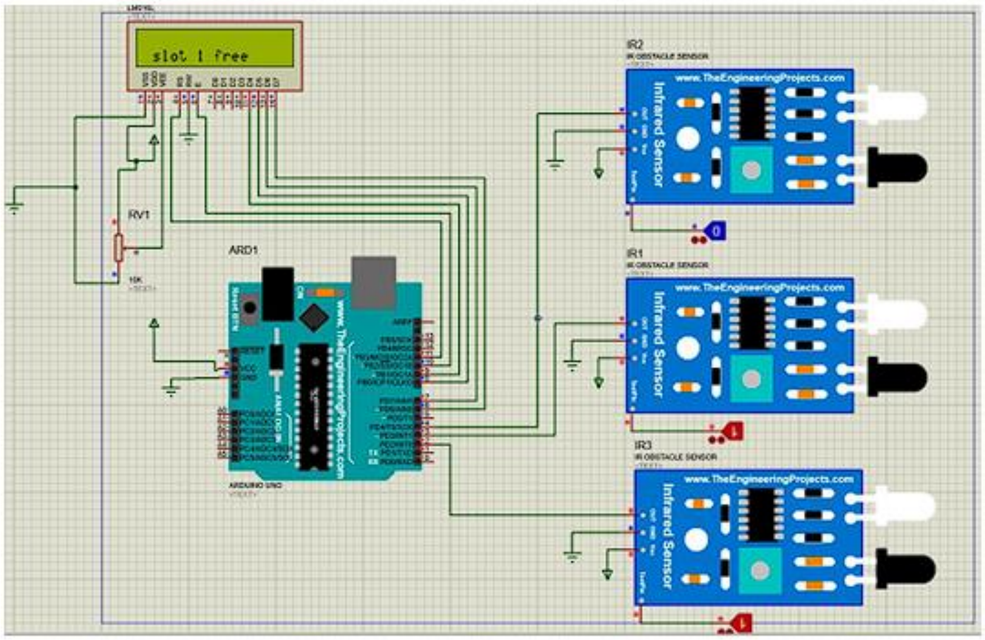
The infrared Obstacle Sensor Module has a built-in IR transmitter and IR receiver that sends out IR energy and looks for reflected IR energy to detect the presence of any obstacle in front of the sensor module. IR sensor is an electronic device, that emits the light in order to sense some object of the 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.