Department of Computer Science and Engineering

Lab Manual

for

Embedded Systems and IOT

CS3691

Regulation 2021

CS3691 – VI Semester

ADHIPARASAKTHI ENGINEERING COLLEGE

Melmaruvathur, Chengalpet District- 603319.

TABLE OF CONTENTS

Ex. No	Title of Experiments	Page No.
1	8051 Assembly Language Experiments Using Simulator.	2
2	Test Data Transfer Between Registers and Memory	8
3	Perform ALU Operations	11
4	Write Basic Programs Using Embedded C	14
5	Arithmetic programs using embedded C	16
6	Introduction to the Arduino Platform	18
7	Introduction to Arduino Programming	31
8	Different Communication Methods with IoT Devices	33
9	Different Communication Methods - BLUETOOTH Communication	35
10	Different Communication Methods – ZIGBEE Communication	37
11	Different Communication Methods - GSM	41
12	Introduction to the Raspberry Pi Platform	43
13	Introduction To Python Programming	47
14	Interfacing Sensors with Raspberry Pi	52
15	Communicate Between Arduino and Raspberry PI	55
16	Cloud Platform to Log the Data	58
17	Log Data Using Rasberry PI and Upload it to the Cloud Platform	63
18	Design an IoT-based system	65

APEC

Ex. No:1	8051 ASSEMBLY LANGUAGE EXPERIMENTS USING SIMULATOR

AIM:

To write 8051 Assembly Language Program for an 8-bit addition using Keil simulator and execute it.

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

INTRODUCTION TO 8051 SIMULATORS:

A simulator is software that will execute the program and show the results exactly to the program running on the hardware, if the programmer finds any errors in the program while simulating the program in the simulator, he can change the program and re-simulate the code and get the expected result, before going to the hardware testing. The programmer can confidently dump the program in the hardware when he simulates his program in the simulator and gets the expected results.

8051 controller is a most popular 8-bit controller which is used in a large number of embedded applications and many programmers write programs according to their application. So testing their programs in the software simulators is a way. Simulators will help the programmer to understand the errors easily and the time taken for the testing is also decreased. These simulators are very useful for students because they do need not to build the complete hardware for testing their program and validate their program very easily in an interactive way.

List of 8051 Simulators:

The list of simulators is given below with their features:

1. MCU 8051: MCU 8051 is an 8051 simulator that is very simple to use and has an interactive IDE (Integrated Development Environment). It is developed by Martin Osmera and most important of all is that it is completely free. There are many features for this IDE they are

✓ It supports both C and assembly language for compilation and simulation

✓ It will support certain electronic simulations like LED, 7segment display, LCD etc. which will help in giving the output when you interface these things to the hardware directly. It has an in-built source code editor, graphical notepad, ASCII charts, Assembly symbol viewer, etc. It also supports several 8051 ICs like at89c51, A89S52, 8051, 8052, etc.

✓ It has tools like hex decimal editors, base converters, special calculator, file converters, inbuilt hardware programmers, etc.

✓ It has syntax validation, pop base auto-completion etc.

2. EDSIM 51: This is a virtual 8051 interfaced with virtual peripherals like 7 segment display, motor, keypad, UART etc. This simulator is exclusively for students developed by James Rogers,.

The features of this simulator are

✓ Have virtual peripherals like ADC, DAC with scope to display, comparator etc.

✓ Supports only assembly language

✓ IDE is completely written in JAVA and supports all the OS.

✓ Completely free and with user guide, examples, etc.

You can download this simulator from the https://www.edsim51.com/index.html.

3. 8051 IDE: This simulation software is exclusively for the Windows operating system (98/xp).

The features of this simulator are

✓ Text editor, assembler, and software simulate in one single program. ✓ Has facilities like Breakpoint setter, execute to break point, predefined simulator watch window, etc.

✓ It is available in both free version and paid version.

4. KEIL µVision: KEIL is the most popular software simulator. It has many features like interactive IDE and supports both C and assembly languages for compilation and simulation.

INSTALLATION OF KEIL SOFTWARE

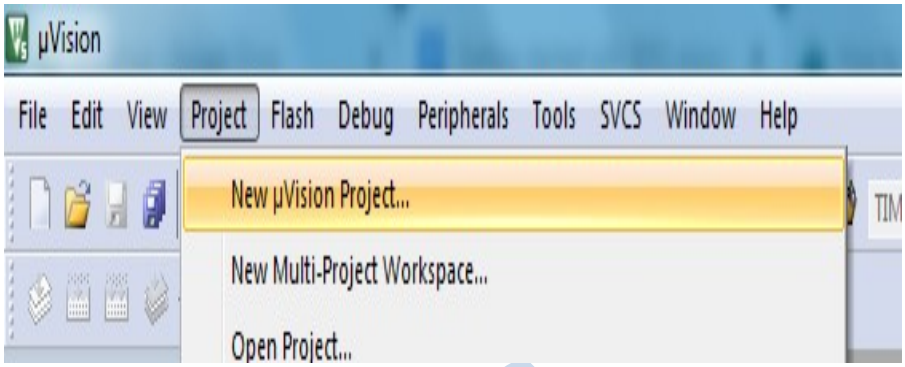
Keil µVision IDE is a popular way to program MCUs containing the 8051 architectures. It supports over 50 microcontrollers and has good debugging tools including logic analysers and watch windows

In this article, we will use the AT89C51ED2 microcontroller, which has:

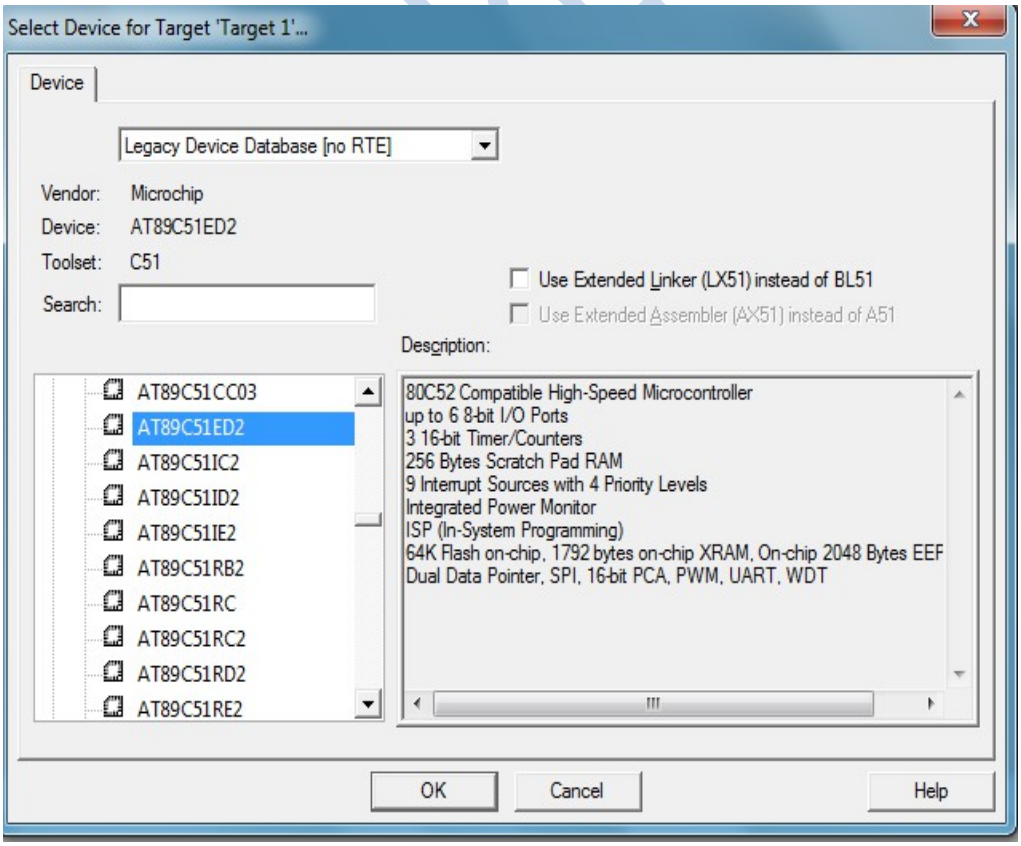
- 64 KB FLASH ROM
- On-chip EEPROM
- 256 Bytes RAM
- In-System programming for uploading the program
- 3 Timer/counters
- SPI, UART, PWM



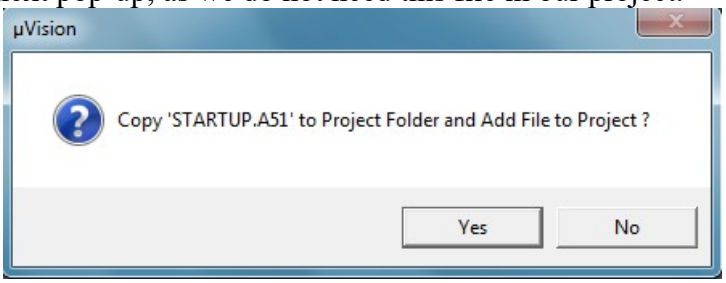
To start writing a new program, you need to create a new project. Navigate to **project —> New µVision project**. Then save the new project in a folder.



After saving the file, a new window will pop up asking you to select your microcontroller. As discussed, we are using AT89C51/AT89C51ED2/AT89C52, so select this controller under the Microchip section (as Atmel is now a part of Microchip).



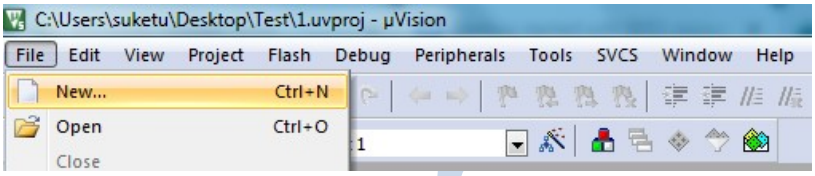
Select '**Yes**' in the next pop-up, as we do not need this file in our project.



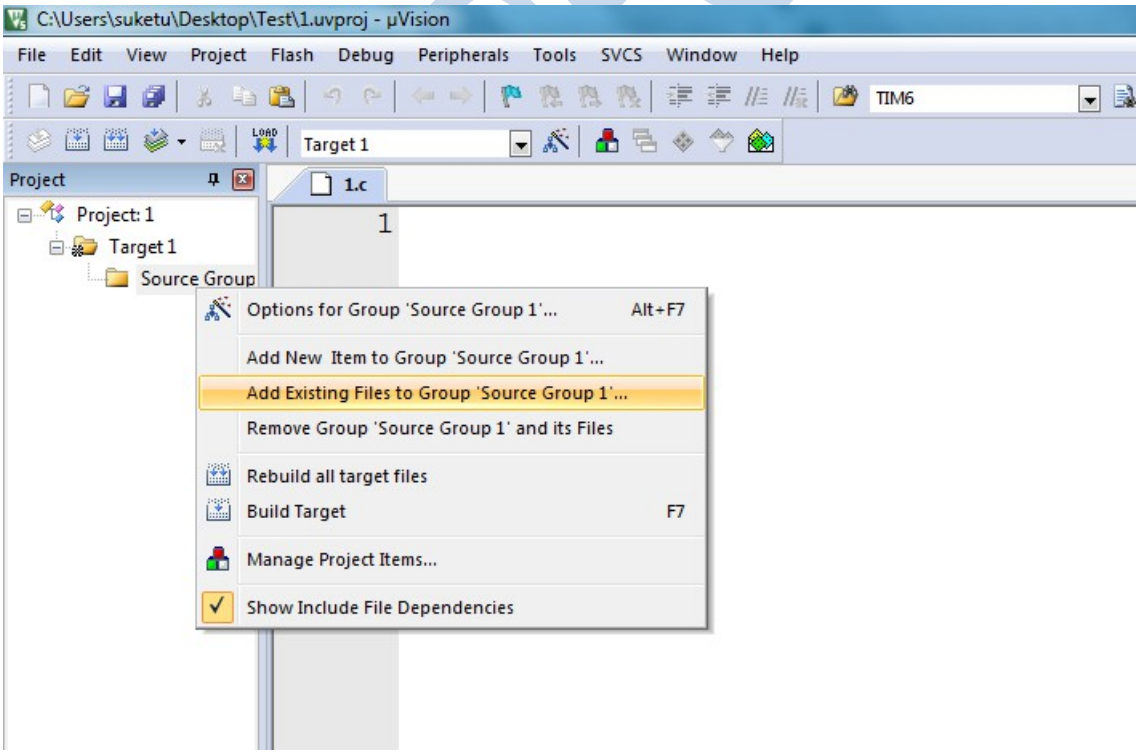
Our project workspace is now ready!

From here, we need to create a file where we can write our C code. Navigate to **File —> New**.

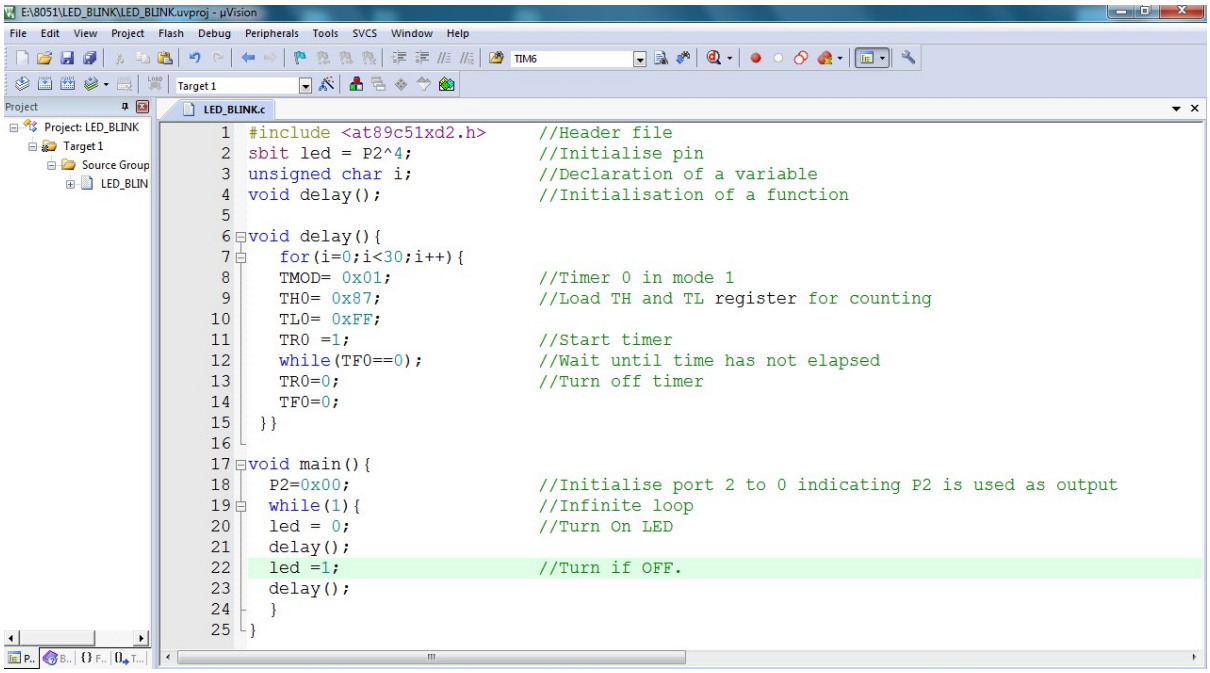
Once the file is created, save it with .c extension in the same project folder.



Next, we have to add that .c or .asm file to our project workspace. Select **Add Existing** Files and then select the created .c or .asm file to get it added.



The workspace and project file are ready.



PROCEDURE

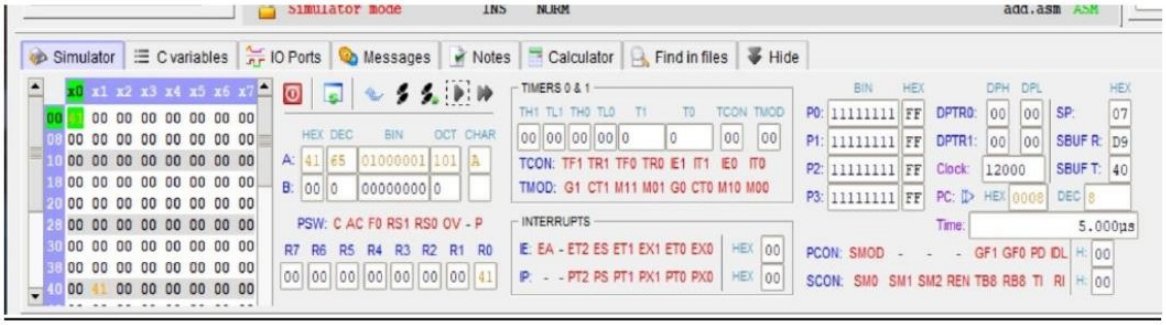
1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µVision Project and Create New Project Select Device for Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of simulation by clicking Run or step run.8.

PROGRAM:

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

APEC

OUTPUT:



APEC

RESULT:

Thus the 8051 Assembly language for an 8-bit Addition using Keil Simulator is executed successfully

Ex.No:2	TEST DATA TRANSFER BETWEEN REGISTERS AND MEMORY

AIM:

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

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

PROCEDURE

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µVision Project and Create a New Project Select Device for the Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of the simulation by clicking Run or Step run.

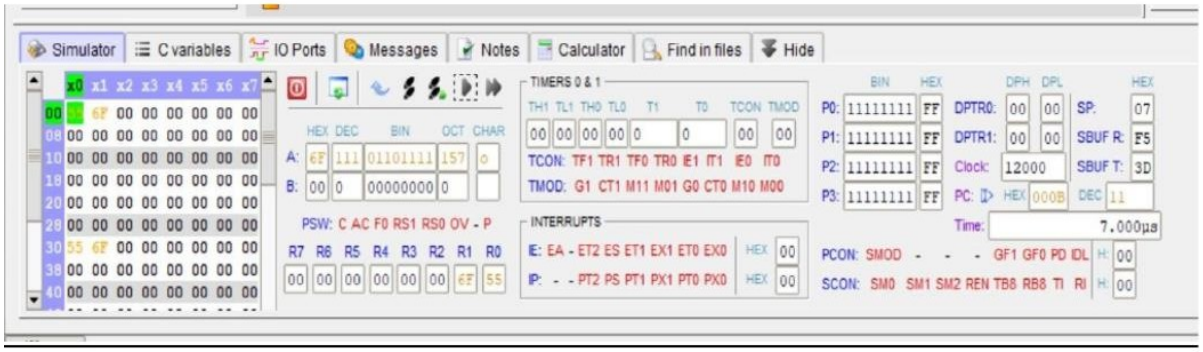
PROGRAM:

TYPE-I:

```
ORG 0000H
CLR C
MOV R0, #55H
MOV R1, #6FH
MOV A, R0
MOV 30H, A
MOV A, R1
MOV 31H, A
END
```

APEC

OUTPUT:



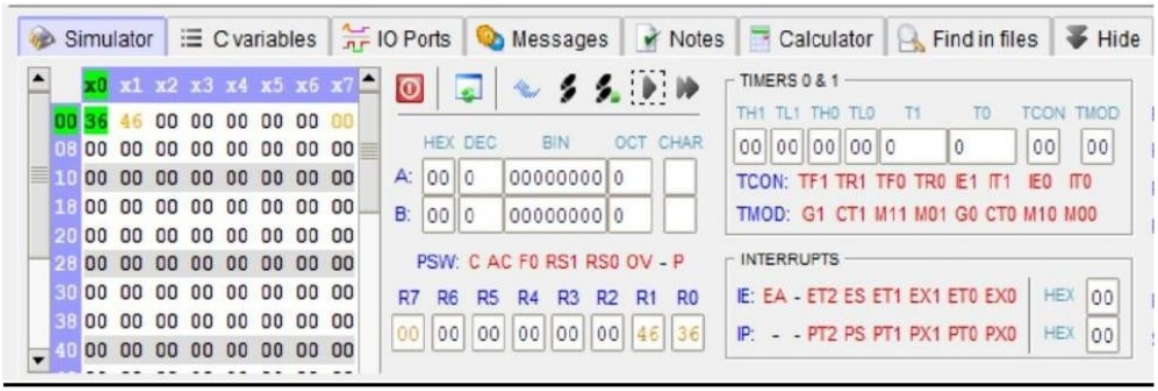
PROGRAM:

TYPE-II:

```
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
```

APEC

OUTPUT:



APEC

RESULT:

Thus, the Assembly language Program to transfer data between register and memory was executed successfully

Ex. No:3	PERFORM ALU OPERATIONS

AIM:

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

SOFTWARE TOOLS REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

PROCEDURE:

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µVision Project and Create New Project Select Device for Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of simulation by clicking Run or step run.

PROGRAM:

```
ORG 0000H
CLR C

//ADDITION
MOV A, #20H
ADD A, #21H
MOV 41H, A

//SUBTRACTION
MOV A, #20H
SUBB A, #18H
```

APEC

MOV 42H, A

//MULTIPLICATION

MOV A, #03H

MOV B, #04H

MUL AB

MOV 43H, A

//DIVISION

MOV A, #95H

MOV B, #10H

DIV AB

MOV 44H, A

MOV 45H, B

//AND

MOV A, #25H

MOV B, #12H

ANL A, B

MOV 46H, A

//OR

MOV A, #25H

MOV B, #15H

ORL A, B

MOV 47H, A

//XOR

MOV A, #45H

MOV B, #67H

XRL A, B

MOV 48H, A

APEC

//NOT

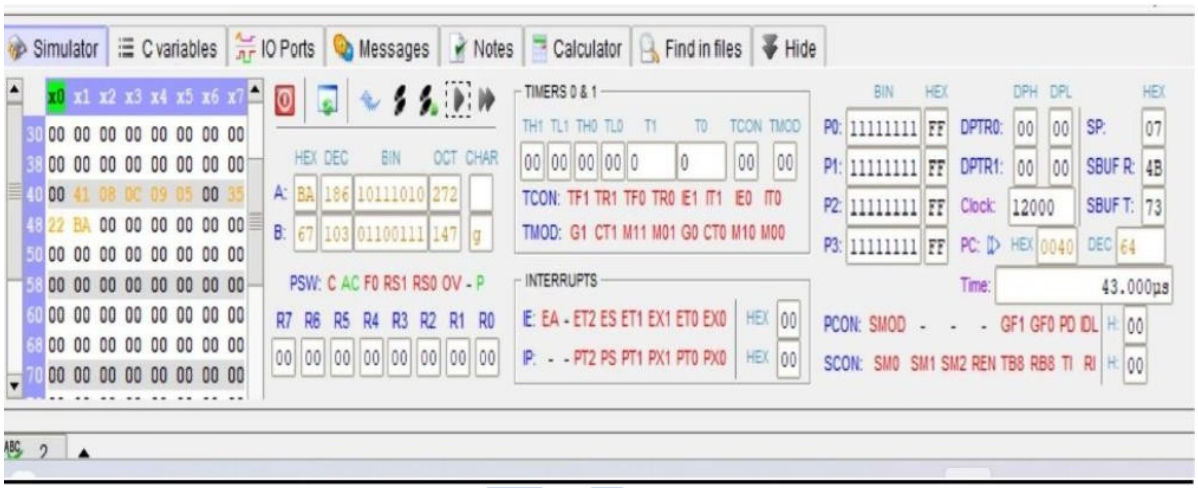
MOV A, #45H

CPL A

MOV 49H, A

END

OUTPUT:



RESULT:

Thus, we executed the Perform ALU operations

Ex. No:4	WRITE BASIC PROGRAMS USING EMBEDDED C

AIM:

To write a basic embedded C program to control a port 0 pin 0 connected to an 8051-microcontroller using a Keil simulator.

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

PROCEDURE

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µVision Project and Create a New Project Select Device for the Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of the simulation by clicking Run or Step run.

PROGRAM:

```
#include<REG51.h>
int i,j;
sbit LED = P2^0;
void main()
{
while(1)
{
LED = 0;
```

```
for(j=0;j<10000;j++);
LED = 1;
for(j=0;j<10000;j++);
}
}
```

APEC

RESULT:

Thus, the Basic and arithmetic Programs Using Embedded C was Executed successfully.

Ex. No:5	ARITHMETIC PROGRAMS USING EMBEDDED C

AIM:

To write an embedded C program for addition, subtraction, multiplication, and division using the Keil simulator.

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil µvision5 IDE	1

PROCEDURE:

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µvision Project and Create New Project Select Device for Target.
3. Select the device AT89C51ED2 or AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with the .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of the simulation by clicking Run or Step run.

PROGRAM:

```
#include<REG51.H>
unsigned char a, b;
void main()
{ a=0x1
0;
b=0x04;
P0=a-b;
P1=a+b;
P2=a*b;
P3=a/b;
while(1);
}
```

APEC

OUTPUT:

Port P0= _____

Port P1= _____

Port P2= _____

Port P3= _____

APEC

RESULT:

Thus, the embedded c program for Addition, Subtraction, Multiplication, and Division was executed successfully.

Ex. No:6	INTRODUCTION TO THE ARDUINO PLATFORM

AIM:

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

Hardware & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Arduino IDE 2.0	1
2	Arduino Uno Board	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.

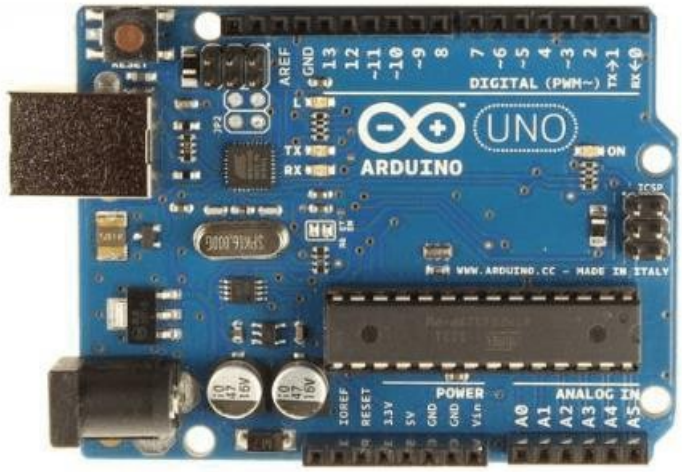
The various components present on the Arduino boards are a Microcontroller, Digital Input/output pins, USB Interface and Connector, Analog Pins, reset buttons, Power buttons, LEDs, Crystal oscillators, and Voltage regulators. Some components may differ depending on the type of board. The most standard 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 analog 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 mini-USB. The Arduino Nano includes an I/O pin set of 14 digital pins and 8 analog pins. It also includes6 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. TheArduino Mega includes 54 I/O digital pins and 16 Analog Input/Output (I/O), ICSP header, a reset button, 4 UART (Universal Asynchronous Reciever/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 analog 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

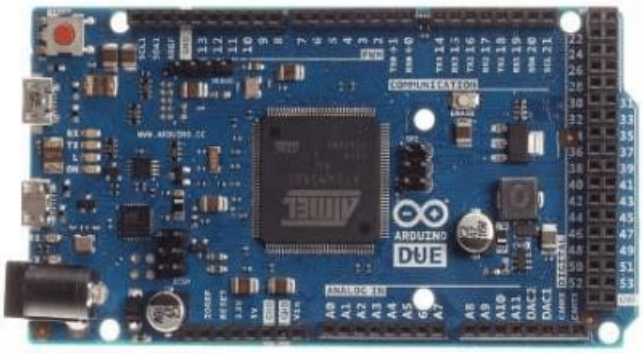
The basic specification of the Arduino Leonardo is the same as the Arduino Micro. It is also based on the ATmega32U4 Microcontroller. The components present on the board are 20 analog

and digital pins, a reset button, a 16MHz crystal oscillator, an ICSP header, and a micro USB connection.



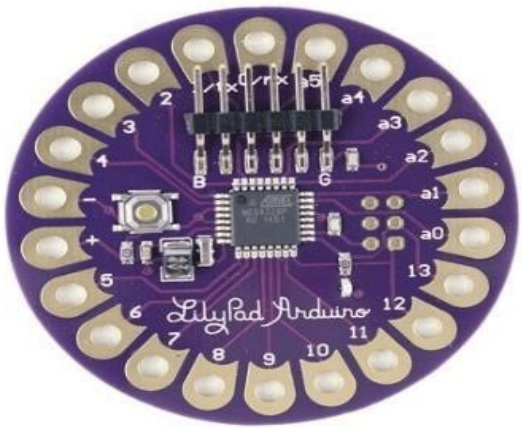
6) Arduino Due

The Arduino Due is based on the 32-bit ARM core. It is the first Arduino board that has been developed based on the ARM Microcontroller. It consists of 54 Digital Input/Output pins and 12 Analog pins. The Microcontroller present on the board is the Atmel SAM3X8E ARM Cortex-M3 CPU. It has two ports, namely, a native USB port and a Programming port. The micro side of the USB cable should be attached to the programming port.



7) Arduino Lilypad

The Arduino LilyPad was initially created for wearable projects and e-textiles. It is based on the ATmega168 Microcontroller. The functionality of Lilypad is the same as other Arduino Boards. It is a round, lightweight board with a minimal number of components to keep the size of the board small. The Arduino Lilypad board was designed by Sparkfun and Leah. It was developed by Leah Buechley. It has 9 digital I/O pins.



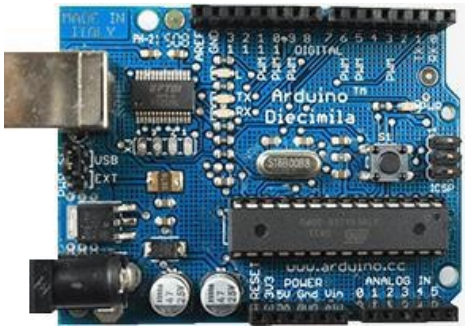
8) Arduino Bluetooth

The Arduino Bluetooth board is based on ATmega168 Microcontroller. It is also named as **Arduino BT board**. The components present on the board are 16 digital pins, 6 analog pins, reset button, 16MHz crystal oscillator, ICSP header, and screw terminals. The screw terminals are used for power. The Arduino Bluetooth Microcontroller board can be programmed over the Bluetooth as a wireless connection.



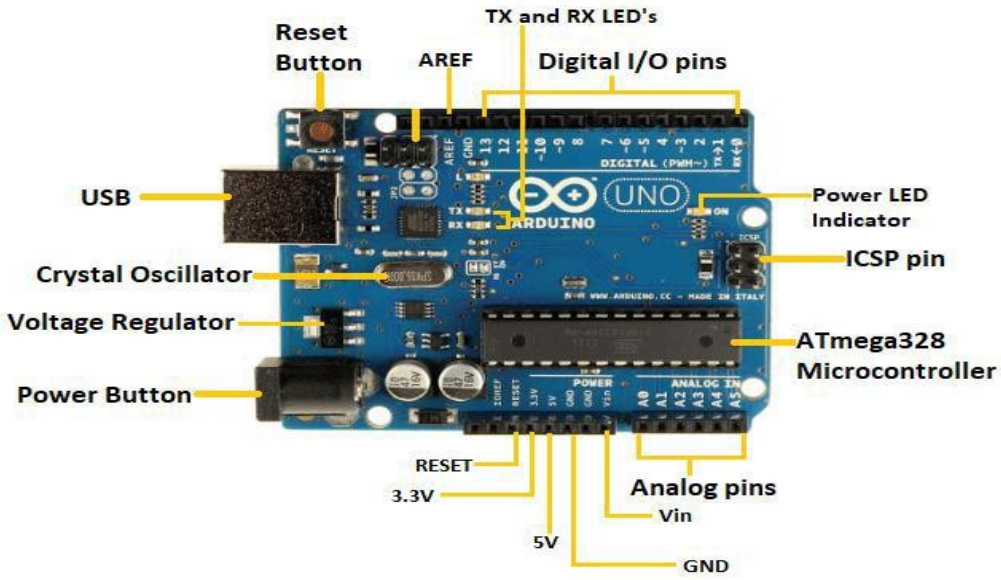
9) Arduino Diecimila

The Arduino Diecimila is also based on the ATmeg628 Microcontroller. The board consists of 6 analog pin inputs, 14 digital Input/Output pins, a USB connector, a power jack, an ICSP (In-Circuit Serial Programming) header, and a reset button. We can connect the board to the computer using the USB and can power on the board with the help of an AC to DC adapter. The Diecimila was initially developed to mark the 10000 delivered boards of Arduino. Here, Diecimila means 10,000 in Italian.



INTRODUCTION TO ARDUINO UNO:

The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered a powerful board used in various projects. Arduino. cc developed the Arduino UNO board. 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 board consists of digital and analog Input/Output pins (I/O), shields, and other circuits. The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms. The IDE is common to all available boards of Arduino.



INTRODUCTION TO ARDUINO IDE 2.0:

The Arduino IDE 2.0 is an open-source project, currently in its beta-phase. It is a big step from its sturdy predecessor, Arduino IDE 2.0, and comes with revamped UI, improved board & library manger, autocomplete feature and much more.

In this tutorial, we will go through step by step, how to download and install the software.

Download the editor

Downloading the Arduino IDE 2.0 is done through the Arduino Software page. Here you will also find information on the other editors available to use.

Requirements

• **Windows** - Win 10 and newer, 64 bits

• **Linux** - 64 bits

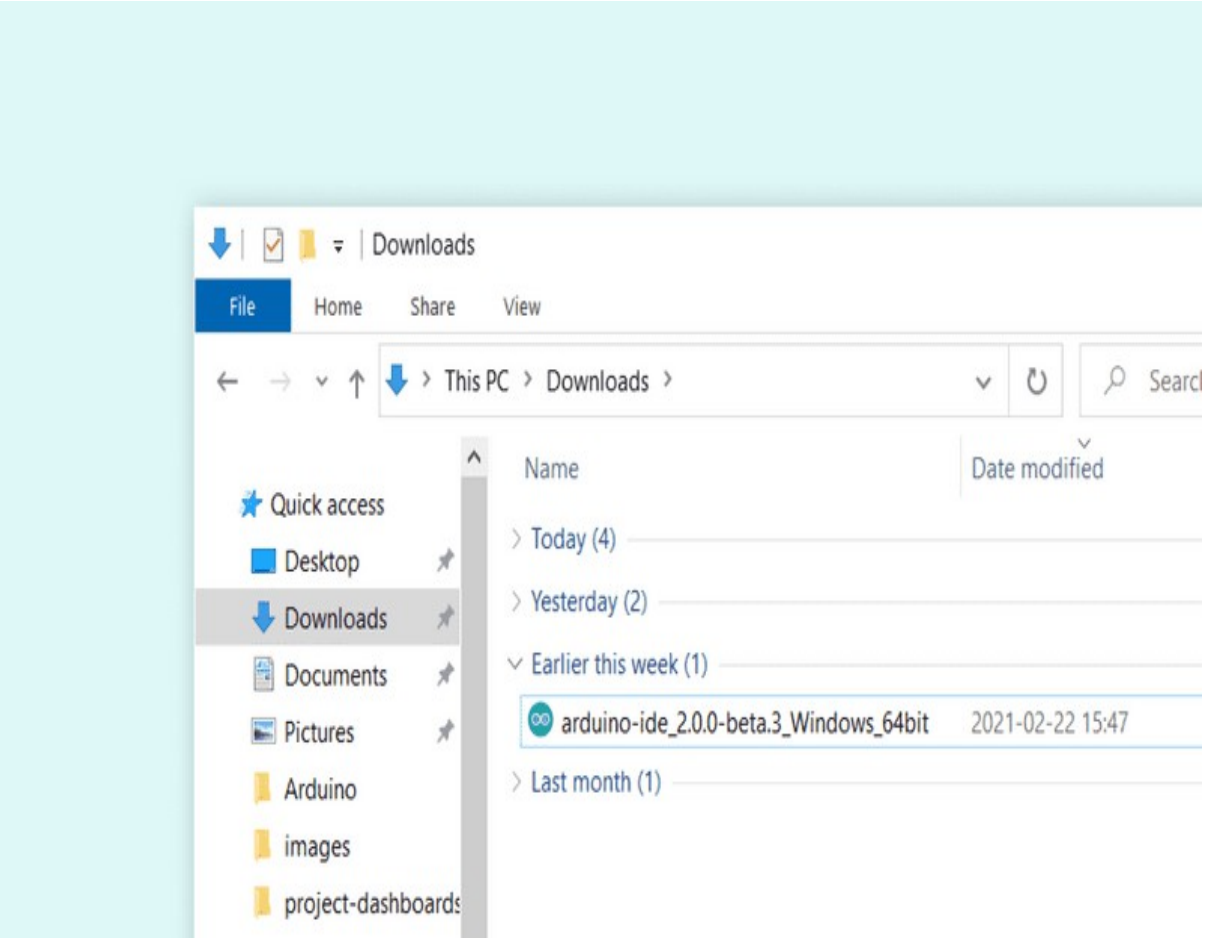
• **Mac OS X** - Version 10.14: "Mojave" or newer, 64 bits

Installation

Windows

Download URL: https://www.arduino.cc/en/software

To install the Arduino IDE 2.0 on a Windows computer, simply run the file downloaded from the software page.



How to use the board manager with the Arduino IDE 2.0

The board manager is a great tool for installing the necessary cores to use your Arduino boards. In this quick tutorial, we will take a look at how to install one, and choosing the right core for your board!

Requirements

• Arduino IDE 2.0 installed.

Why use the board manager?

The board manager is a tool that is used to install different cores on your local computer. So what is a **core**, and why is it necessary that I install one?

Simply explained, a core is written and designed for specific microcontrollers. As Arduino have several different types of boards, they also have different type of microcontrollers.

For example, an Arduino UNO has an **ATmega328P**, which uses the **AVR core**, while an Arduino Nano 33 IoT has a **SAMD21** microcontroller, where we need to use the **SAMD core**.

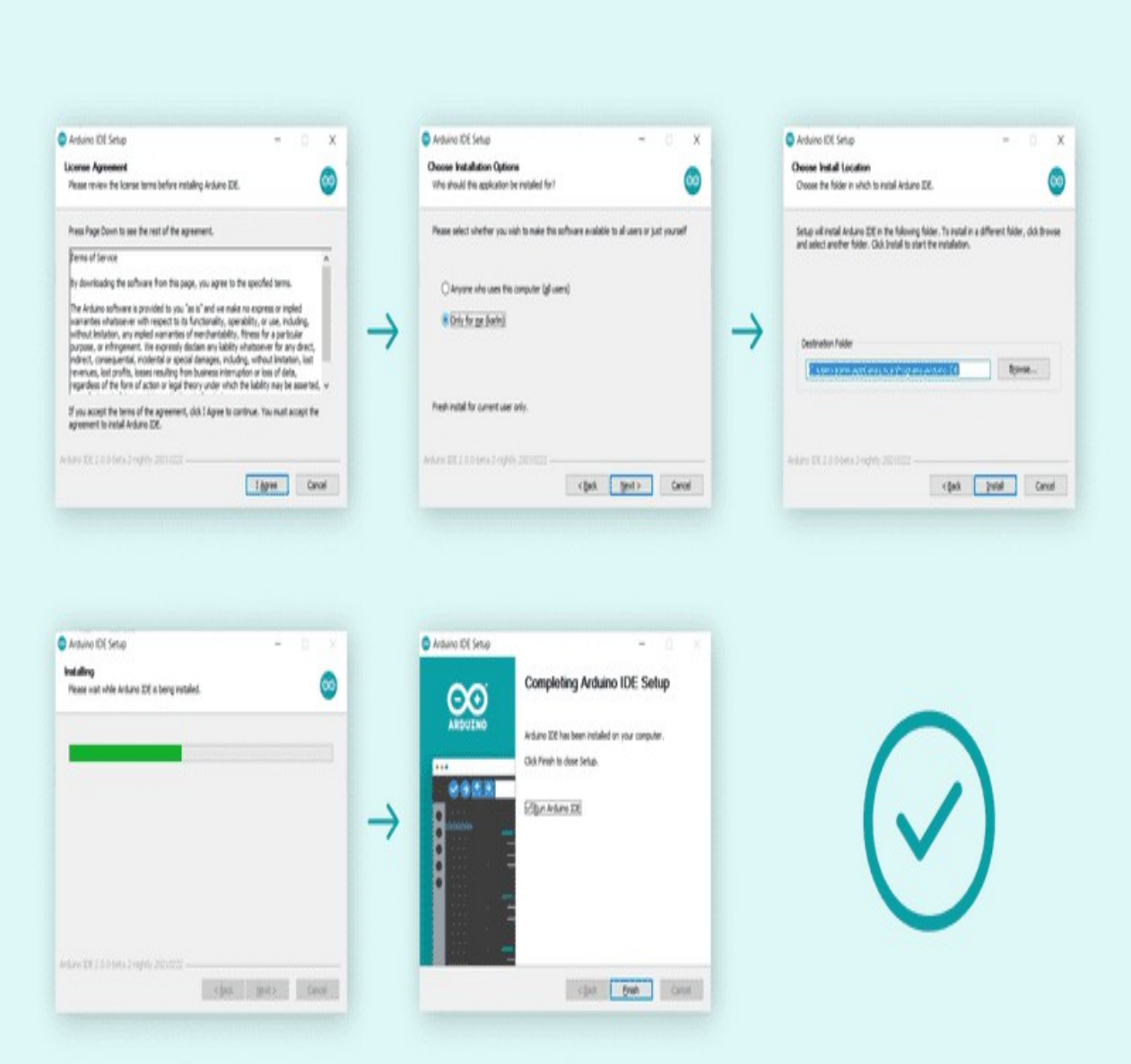
In conclusion, to use a specific board, we need to install a specific core.

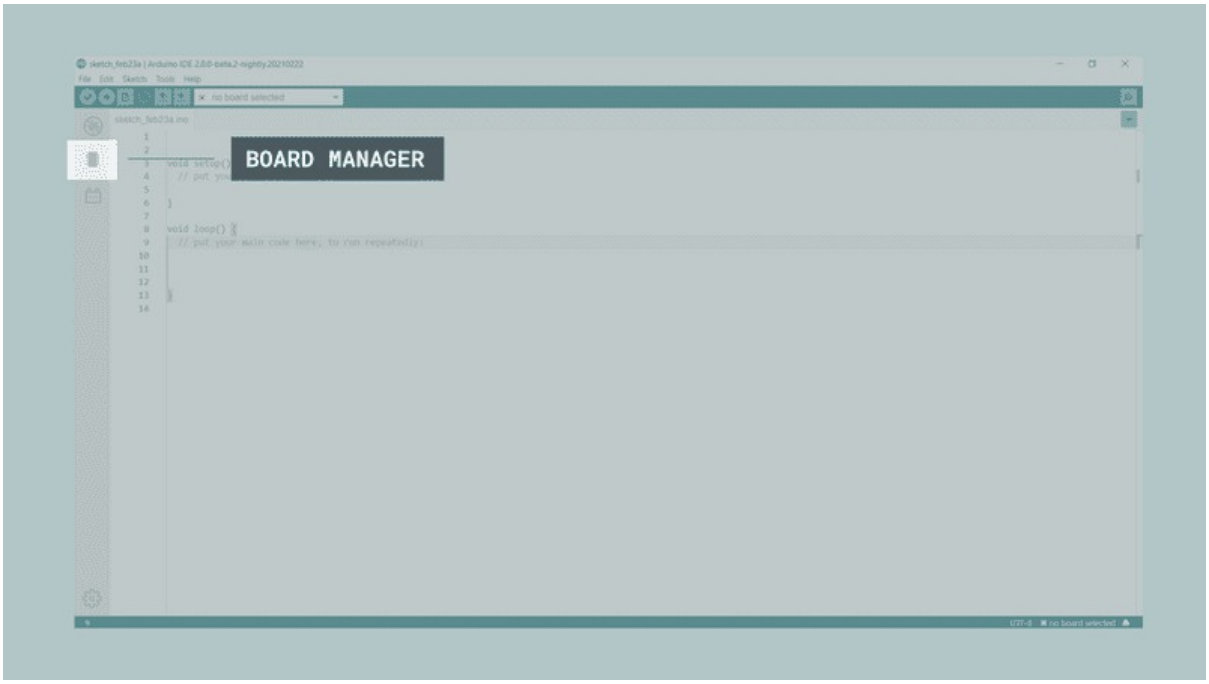
Installing a core

Installing a core is quick and easy, but let's take a look at what we need to do.

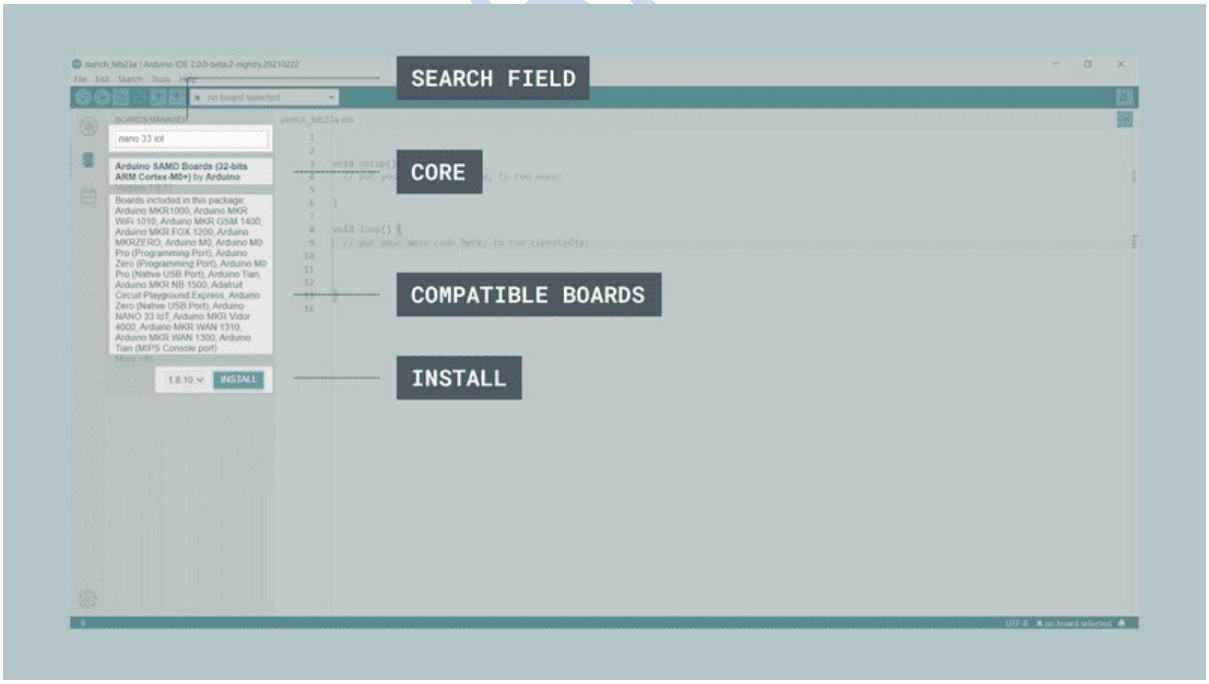
1. Open the Arduino IDE 2.0.

2. With the editor open, let's take a look at the left column. Here, we can see a couple of icons. Let's click the on the **"computer chip"** icon.

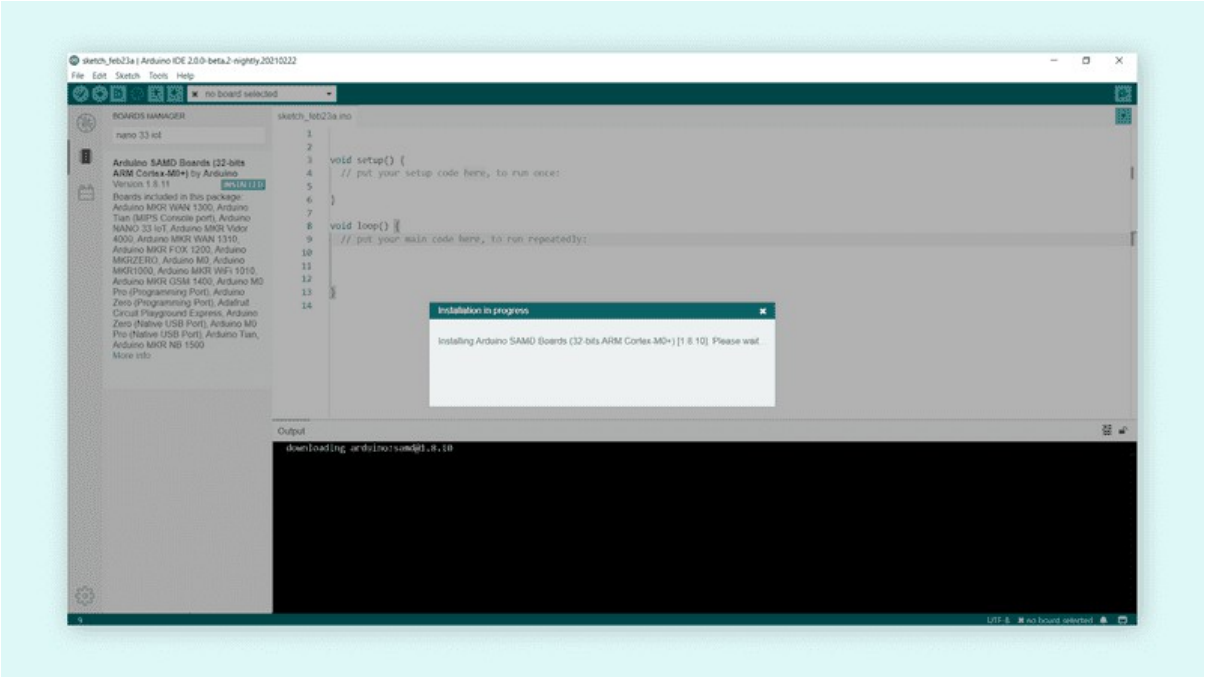




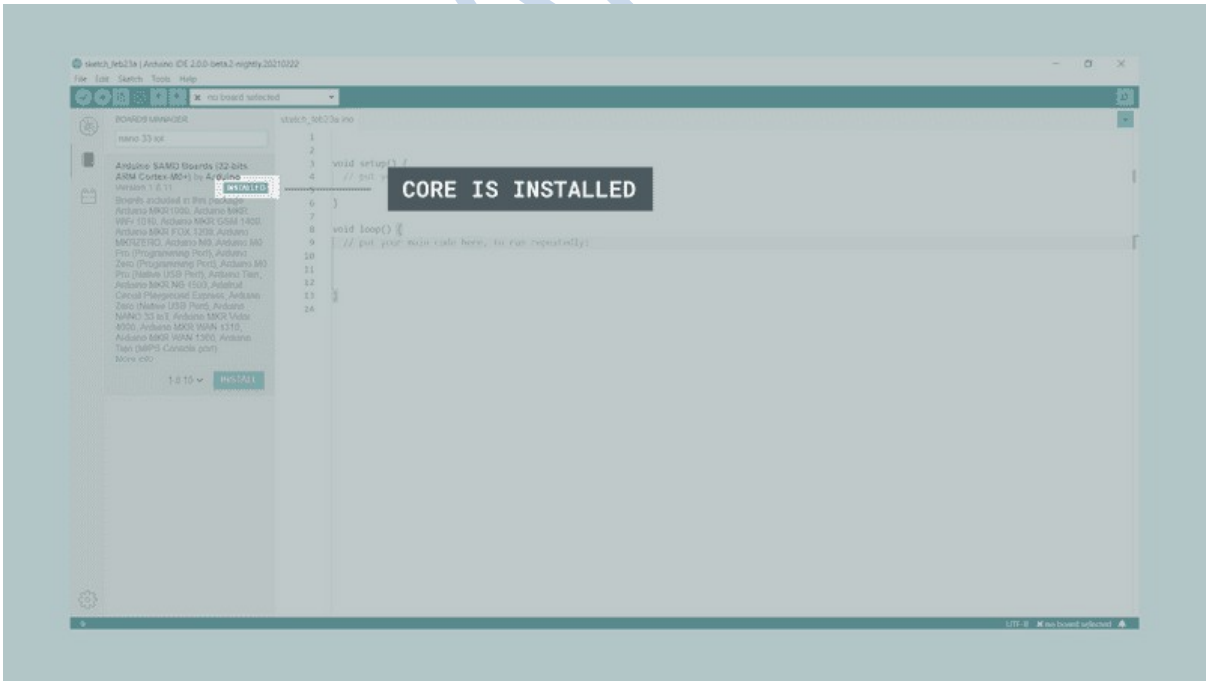
A list will now appear of all available cores. Now let's say we are using an **Nano 33 IoT** board, and we want to install the core. Simply enter the name in the search field, and the right core (SAMD) will appear, where the Nano 33 IoT features in the description. Click on the **"INSTALL"** button.



This will begin an installation process, which in some cases may take several minutes.



When it is finished, we can take a look at the core in the boards manager column, where it should say **"INSTALLED"**.



You have now successfully downloaded and installed a core on your machine, and you can start using your Arduino board!

How to upload a sketch with the Arduino IDE 2.0

In the Arduino environment, we write **sketches** that can be uploaded to Arduino boards. In this tutorial, we will go through how to select a board connected to your computer, and how to upload a sketch to that board, using the Arduino IDE 2.0.

Requirements

• Arduino IDE 2.0 installed.

Verify VS Upload

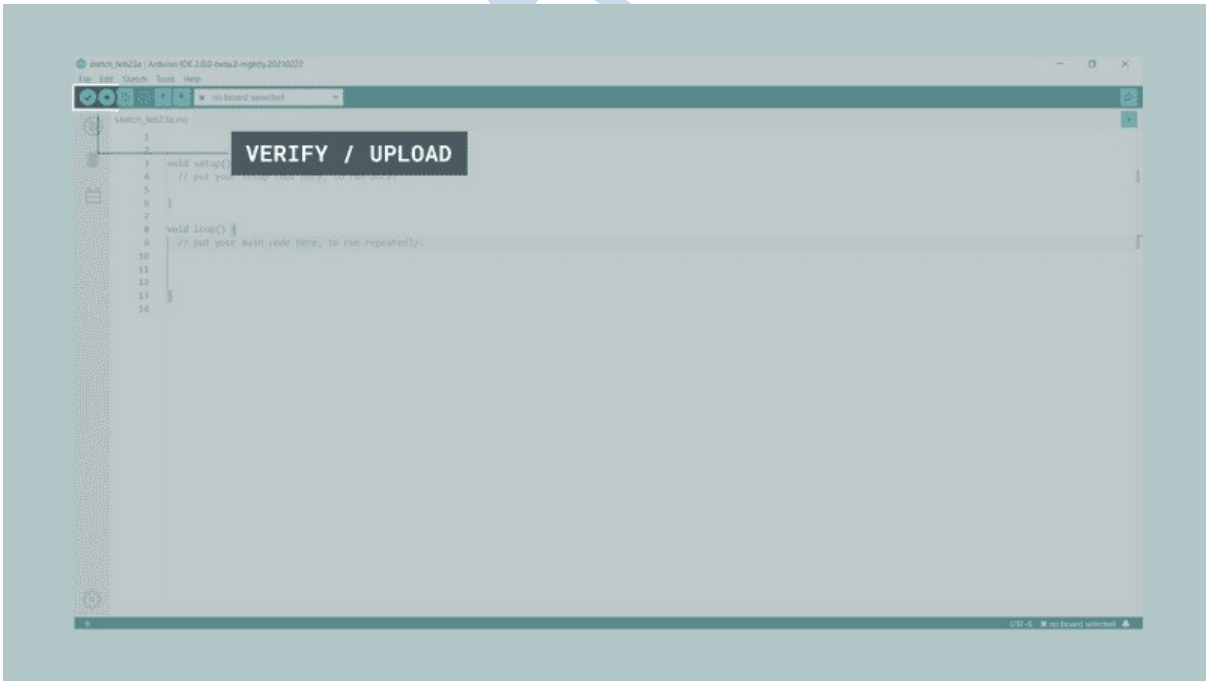
There are two main tools when uploading a sketch to a board: **verify** and **upload.** The verify tool simply goes through your sketch, checks for errors and compiles it. The upload tool does the same, but when it finishes compiling the code, it also uploads it to the board. A good practice is to use the verifying tool before attempting to upload anything. This is a quick way of spotting any errors in your code, so you can fix them before actually uploading the code.

Uploading a sketch

Installing a core is quick and easy, but let's take a look at what we need to do.

1. Open the Arduino IDE 2.0.

2. With the editor open, let's take a look at the navigation bar at the top. At the very left, there is a **checkmark** and an **arrow pointing right**. The checkmark is used to **verify**, and the arrow is used to **upload.**

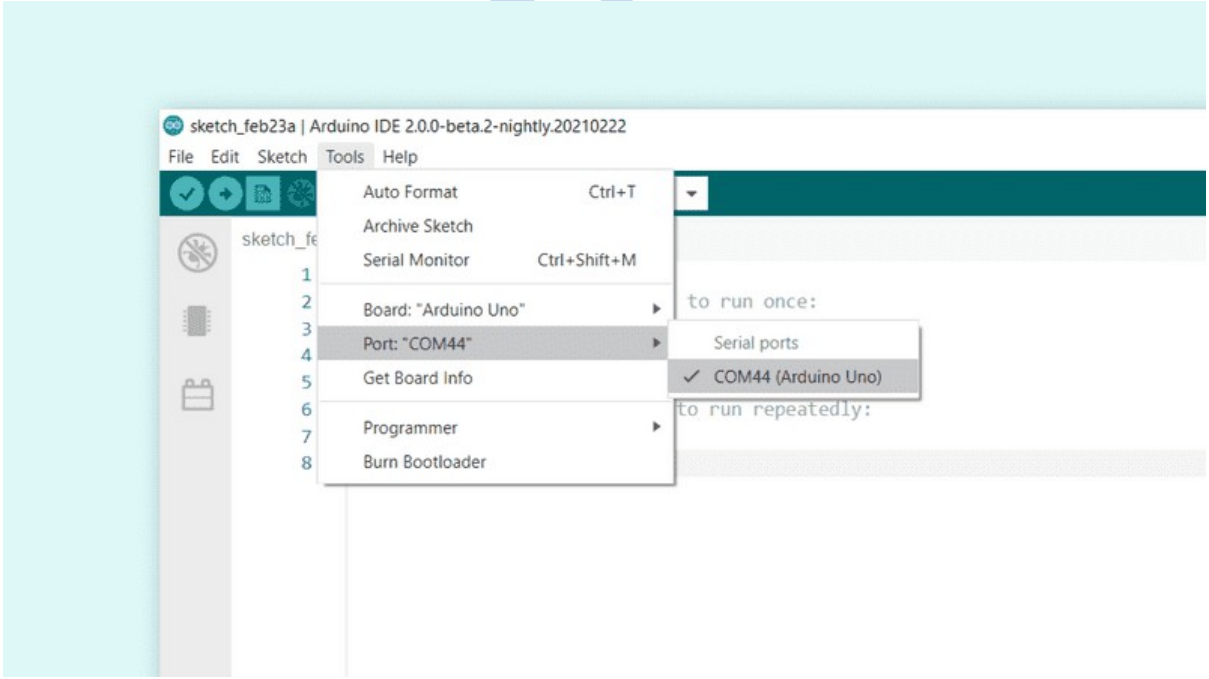


Click on the verify tool (checkmark). Since we are verifying an empty sketch, we can be sure

it is going to compile. After a few seconds, we can see the result of the action in the console (black box in the bottom).



Now we know that our code is compiled, and that it is working. Now, before we can upload the code to our board, we will first need to select the board that we are using. We can do this by navigating to **Tools > Port > {Board}**. The board(s) that are connected to your computer should appear here, and we need to select it by clicking it. In this case, our board is displayed as **COM44 (Arduino UNO)**.



With the board selected, we are good to go! Click on the **upload** button, and it will start uploading the sketch to the board.

When it is finished, it will notify you in the console log. Of course, sometimes there are some complications when uploading, and these errors will be listed here as well.



RESULT:

Thus, the study of Arduino Platform has been done and ensured its composition with internalfeatures specifically.

Ex. No:7	INTRODUCTION TO ARDUINO PROGRAMMING

AIM:

To write and execute different Arduino programming for analog, digital signals and Serial communication.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Arduino IDE 2.0	1
2	Arduino Uno Board	1
3	Jumper Wires	few
4	Arduino USB Cable	1

PROCEDURE:

1. Connect the Arduino Uno board to your computer.
2. Open the Arduino IDE by clicking on 

.
3. Go to "Tools" -> "Port" and select the port to which Uno board is connected.
4. Again go to "Tools" -> "Board" and select Arduino Uno board.
5. Copy the program given below into the IDE and click "Upload".
6. You will observe onboard LED blinking after the given time delay.

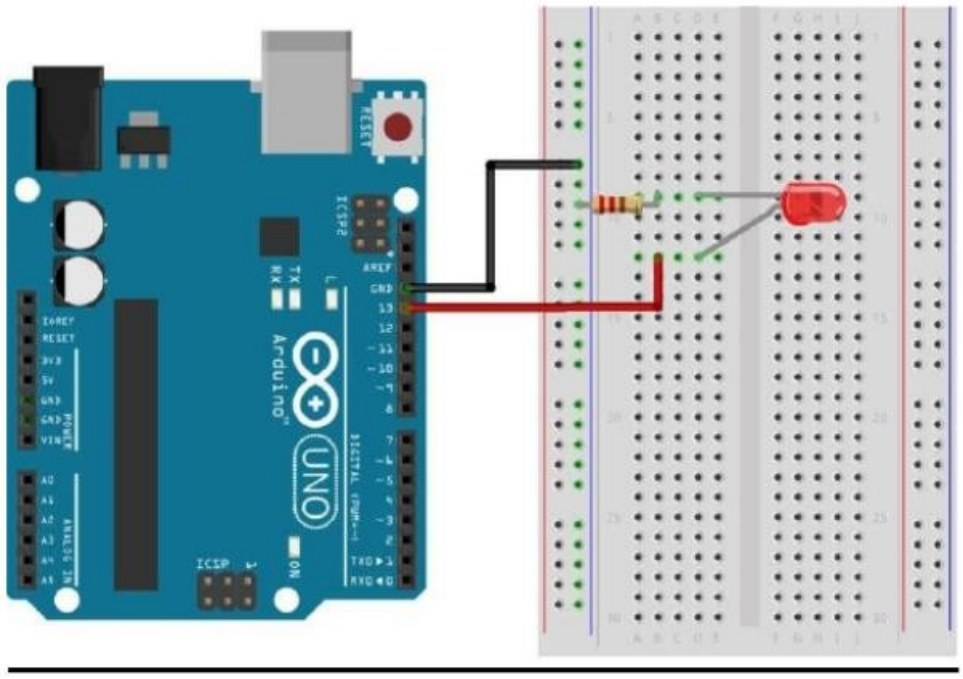
PROGRAM:

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

APEC

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

OUTPUT:



RESULT:

Thus, the LED blinking with Arduino has been verified and observed the output successfully.

Ex. No:8	DIFFERENT COMMUNICATION METHODS WITH IOT DEVICES

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.

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. Factors to consider include range, power consumption, data rate, security, and interoperability with other devices or systems.

APEC

RESULT:

Thus, the study of different communication methods with IOT has been done and ensured its composition with internal features specifically.

Ex. No:9	DIFFERENT COMMUNICATION METHODS - BLUETOOTH COMMUNICATION

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Arduino IDE 2.0	1
2	Arduino UNO Development Board	1
3	Jumper Wires	few
4	Arduino USB Cable	1
5	HC-05 Bluetooth Module	1
6	BlueTooth Seriel Monitor App	1

CONNECTIONS:

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

PROCEDURE:

1. Connect the Arduino Uno board to your computer.
2. Open the Arduino IDE by clicking on 

.
3. Go to "Tools" -> "Port" and select the port to which Uno board is connected.
4. Again go to "Tools" -> "Board" and select Arduino Uno board.
5. Type the program given below into the IDE and click "Upload".
6. Before uploading we will remove the transmitter and receiver in the Arduino board. After completion only we insert the transmitter and receiver pin connections

AEC

7. In the mobile open serial monitor app and then type any message and the message will displays on serial monitor.

PROGRAM:

```
void setup()
{
 Serial.begin(9600);
}
void loop() {
      while (Serial.available())
   {
    String m=Serial.readString();
    Serial.println(m);
   }
   delay(100);
}
```

APEC

RESULT:

Thus, the program to control an LED using a Bluetooth module has been verified and executed successfully.

Ex. No:10	DIFFERENT COMMUNICATION METHODS - ZIGBEE COMMUNICATION

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Arduino IDE 2.0	1
2	Arduino UNO Development Board	2
3	Jumper Wires	few
4	Arduino USB Cable	2
5	Zigbee Module	2

APEC

PROGRAM:

```
#include <stdlib.h>
#include <stdio.h>
int incomingbyte;
byte variable[80];
byte index = 0;
String str, output,output1,output0,output2,output3, result[100];
String m;
long decimal = 0, base = 1;
   int i = 0, value, length;
   int count=0;
//=====================================
//=           SETUP           =
//=====================================
float HexToFloat(String input)
{
  int len = input.length() + 1;
  char hstr[len];
  input.toCharArray(hstr, len);
```

```
  union
    { uint32_t m;
    float f;
  } data;
  data.m = strtoul(hstr, NULL, 16);
  return data.f;
}
int temp;
int temp1;
void setup()
{
  Serial.begin(9600);
  Serial.println("+++ Search ID +++");
}

void loop()
{

  float res1 = 0;
  int res2 = 0;
  while (Serial.available() > 0)
  {
    //Serial.println("coming");
    incomingbyte = Serial.read();
   output = String(incomingbyte, HEX);
  // Serial.println(output);
    if( output == "7e")
    {
     count = 0;
     //Serial.print("variable 0: ");
    // Serial.println(output);
   }
  else
   {
     if(count < 56) // Make sure there is room
     {
      //Serial.println("second");
      result[count++] = output;
```

APEC

```
      count++;
      //Serial.println(output);
     }
    }
   }
  }
  if(count==54)
  {
//  Serial.println("loop");
// Serial.print("variable 1: "); Serial.println(result[34]);
// Serial.print("variable 2: "); Serial.println(result[36]);
//Serial.print("variable 3 "); Serial.println(result[38]);
//Serial.print("variable 4: "); Serial.println(result[40]);
//  Serial.print("variable 5: "); Serial.println(result[42]);
//    Serial.print("variable 6: "); Serial.println(result[44]);
//    Serial.print("variable 7: "); Serial.println(result[46]);
//    Serial.print("variable 8: "); Serial.println(result[48]);
//    Serial.print("variable 9: "); Serial.println(result[50]);
//    Serial.print("variable 10: "); Serial.println(result[52]);
  count=0;
  output0=result[36]+result[38];
  output1=result[40]+result[42];
  output2=result[44]+result[46];
  output3=result[48]+result[50];
char buf0[4],buf1[4],buf2[4],buf3[4];
output0.toCharArray(buf0,(output0.length()+1));
output1.toCharArray(buf1,(output1.length()+1));
output2.toCharArray(buf2,(output2.length()+1));
output3.toCharArray(buf3,(output3.length()+1));
long n0 = strtol(buf0, NULL, 16);
long n1 = strtol(buf1, NULL, 16);
long n2 = strtol(buf2, NULL, 16);
long n3 = strtol(buf3, NULL, 16);
temp = (n3 * 1.2);
temp1                    =                    (temp/10);
Serial.println("==============================");
Serial.print("adc 0: "); Serial.println(n0);
Serial.print("adc 1: "); Serial.println(n1);
Serial.print("adc 2: "); Serial.println(n2);
```

APEC

```
Serial.print("adc 3: "); Serial.println(n3);
Serial.print("Temperature: "); Serial.print(temp1); Serial.println("°C"); //connected in ADC3
  }

}
```

APEC

RESULT:

Thus, the program to control an LED using a Zigbee module has been verified and executed successfully.

Ex. No:11	DIFFERENT COMMUNICATION METHODS - GSM

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Arduino IDE 2.0	1
2	Arduino UNO Development Board	1
3	Jumper Wires	few
4	Arduino USB Cable	2
5	GSM Module	1

PROGRAM:

MAKE A CALL

```
void setup()
{
  Serial.begin(9600);
  delay(10000);
  Serial.println("ATD +91-10 DIGIT NUMBER;\r");
  delay(1000);
}
void loop()
{

}
```

APEC

SEND SMS

```
void setup()
{
  Serial.begin(9600);
  delay(2000);
  Serial.println("AT+CMGF=1");
  delay(2000);
  Serial.println("AT+CMGS=\"+91-10 DIGIT NUMBER \"\r");
  delay(2000);
  Serial.println("HELLO FROM SIM");
  delay(500);
  Serial.write(26);
}
void loop()
{


}
```

APEC

RESULT:

Thus, the program to control an LED using a GSM has been verified executed successfully.

Ex. No:12	INTRODUCTION TO THE RASPBERRY PI PLATFORM

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.

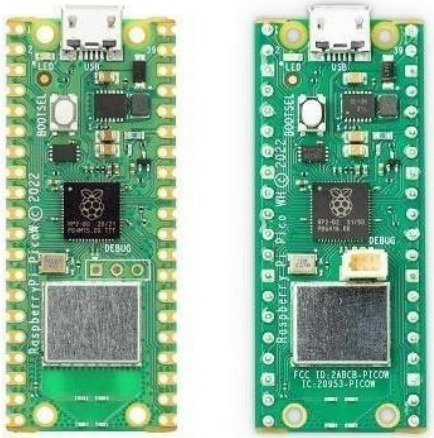
Features:

1. RP2040 microcontroller with 2MB of flash memory
2. On-board single-band 2.4GHz wireless interfaces (802.11n)
3. Micro USB B port for power and data (and for reprogramming the flash)
4. 40 pins 21mmx51mm 'DIP' style 1mm thick PCB with 0.1″ through-hole pins also withedge castellations
5. Exposes 26 multi-function 3.3V general purpose I/O (GPIO)
6. 23 GPIO are digital-only, with three also being ADC-capable
7. Can be surface mounted as a module
8. 3-pin ARM serial wire debug (SWD) port
9. Simple yet highly flexible power supply architecture
10. Various options for easily powering the unit from micro-USB, external supplies, or batteries
11. High quality, low cost, high availability
12. Comprehensive SDK, software examples, and documentation
13. Dual-core Cortex M0+ at up to 133MHz
14. On-chip PLL allows variable core frequency
15. 264kByte multi-bank high-performance SRAM

APEC

Raspberry Pi Pico W:

The Raspberry Pi Pico W is based on the RP2040 microcontroller, which was designed by Raspberry Piin-house. It combines a powerful ARM Cortex-M0+ processor with built-in Wi-Fi connectivity,opening up a range of possibilities for IoT projects, remote monitoring, and wireless communication. The Pico W retains the same form factor as the original Pico, making it compatible with existing Pico accessories and add-ons.



RP2040 Microcontroller:

At the core of the Raspberry Pi Pico W is the RP2040 microcontroller. It features a dual-core ARM Cortex-M0+ processor running at 133MHz, providing ample processing power for a wide range of applications. The microcontroller also includes 264KB of SRAM, which is essential for storing and manipulating data during runtime. Additionally, the RP2040 incorporates 2MB of onboard flash memory for program storage, ensuring sufficient space for your code and firmware.

Wireless Connectivity:

The standout feature of the Raspberry Pi Pico W is its built-in wireless connectivity. It includes an onboard Cypress CYW43455 Wi-Fi chip, which supports dual-band (2.4GHz and 5GHz) Wi-Fi 802.11b/g/n/ac. This allows the Pico W to seamlessly connect to wireless networks, communicate with other devices, and access online services. The wireless capability opens up new avenues for IoT projects, remote monitoring and control, and real-time data exchange.

GPIO and Peripherals:

Similar to the Raspberry Pi Pico, the Pico W offers a generous number of GPIO pins, providing flexibility for interfacing with external components and peripherals. It features 26 GPIO pins, of

which 3 are analog inputs, and supports various protocols such as UART, SPI, I2C, and PWM. The Pico W also includes onboard LED indicators and a micro-USB port for power and data connectivity.

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 syntaxand 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:

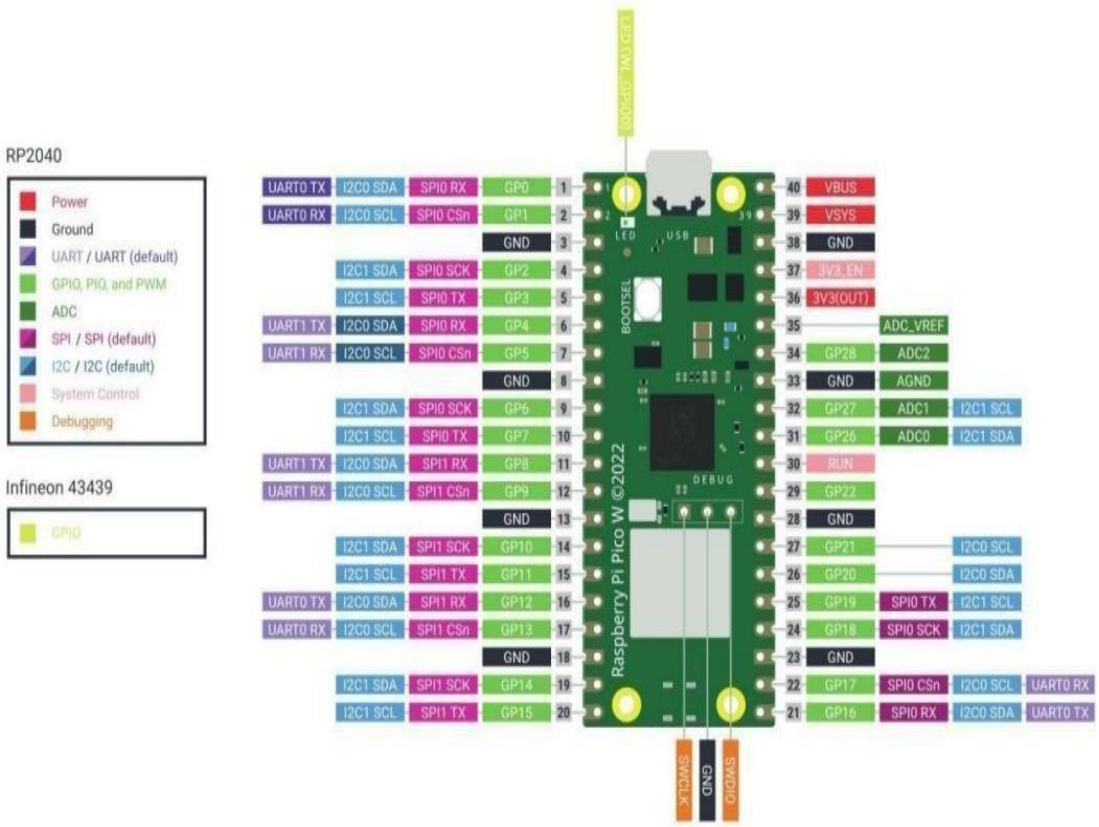
One of the unique features of the RP2040 microcontroller is the inclusion of programmable Input/Output (PIO) state machines. These state machines provide additional processing power and flexibility for handling real-time data and timing-critical applications. The PIO state machines can be programmed 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.

The Raspberry Pi Pico W brings wireless connectivity to the popular Raspberry Pi Pico microcontrollerboard. With its powerful RP2040 microcontroller, built-in Wi-Fi chip, extensive GPIO capabilities, and compatibility with MicroPython and C/C++ programming, the Pico W offers a versatile and affordable platform for a wide range of embedded projects.

APEC



RESULT:

Thus, the study of Raspberry Pi has been done and ensured its composition with internal features specifically.

Ex. No:13	INTRODUCTION TO PYTHON PROGRAMMING

Getting Started with Thonny MicroPython (Python) IDE:

If you want to program your ESP32 and ESP8266 with MicroPython firmware, it's very handy to usean IDE. you'll have your first LED blinking using MicroPython and Thonny IDE.



What is MicroPython?

MicroPython is a Python 3 programming language re-implementation targeted for microcontrollers andembedded systems. MicroPython is very similar to regular Python. Apart from a few exceptions, the language features of Python are also available in MicroPython. The most significant difference betweenPython and MicroPython is that MicroPython was designed to work under constrained conditions.

Because of that, MicroPython does not come with the entire pack of standard libraries. It only includesa small subset of the Python standard libraries, but it includes modules to easily control and interact with the GPIOs, use Wi-Fi, and other communication protocols.

Thonny IDE:

Thonny is an open-source IDE which is used to write and upload MicroPython programs to different development boards such as Raspberry Pi Pico, ESP32, and ESP8266. It is extremely



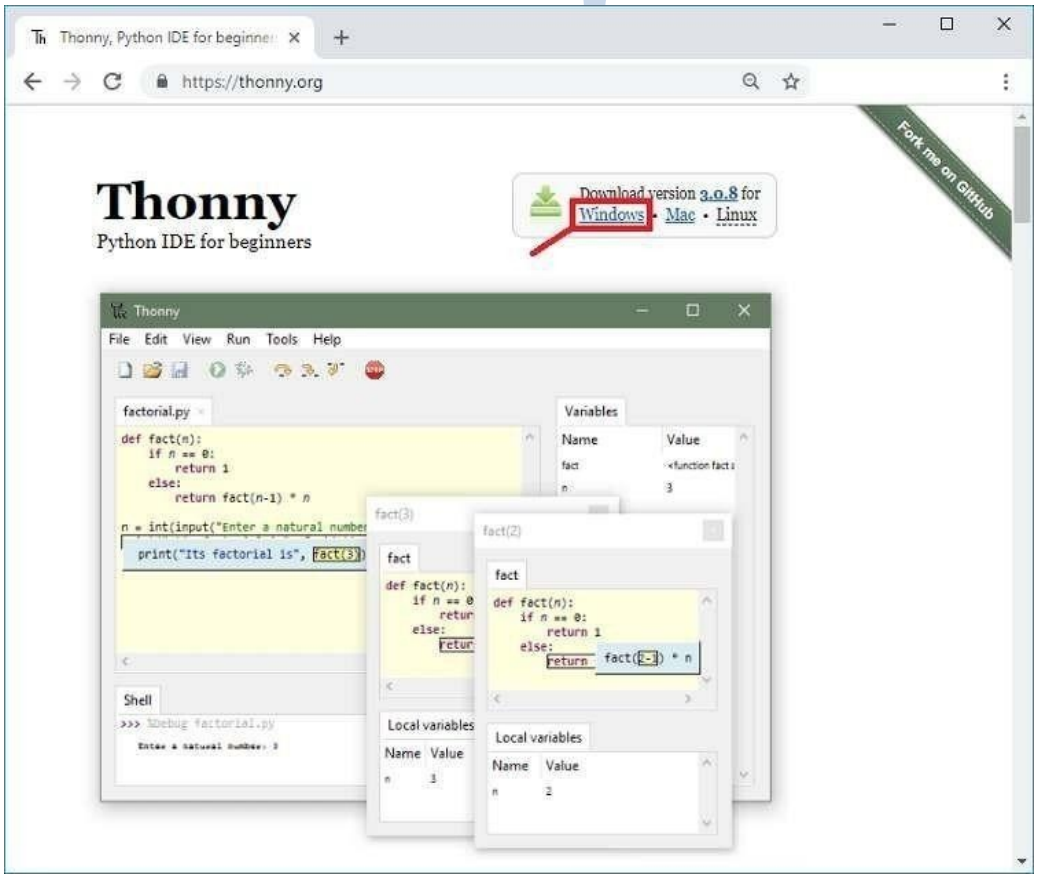
interactive and easy to learn IDE as much as it is known as the beginner-friendly IDE for new programmers. With the help of Thonny, it becomes very easy to code in Micropython as it has a built-in debugger that helps to find any error in the program by debugging the script line by line.

You can realize the popularity of Thonny IDE from this that it comes pre-installed in Raspian OS which is an operating system for a Raspberry Pi. It is available to install on r Windows, Linux, andMac OS.

A) Installing Thonny IDE – Windows PC

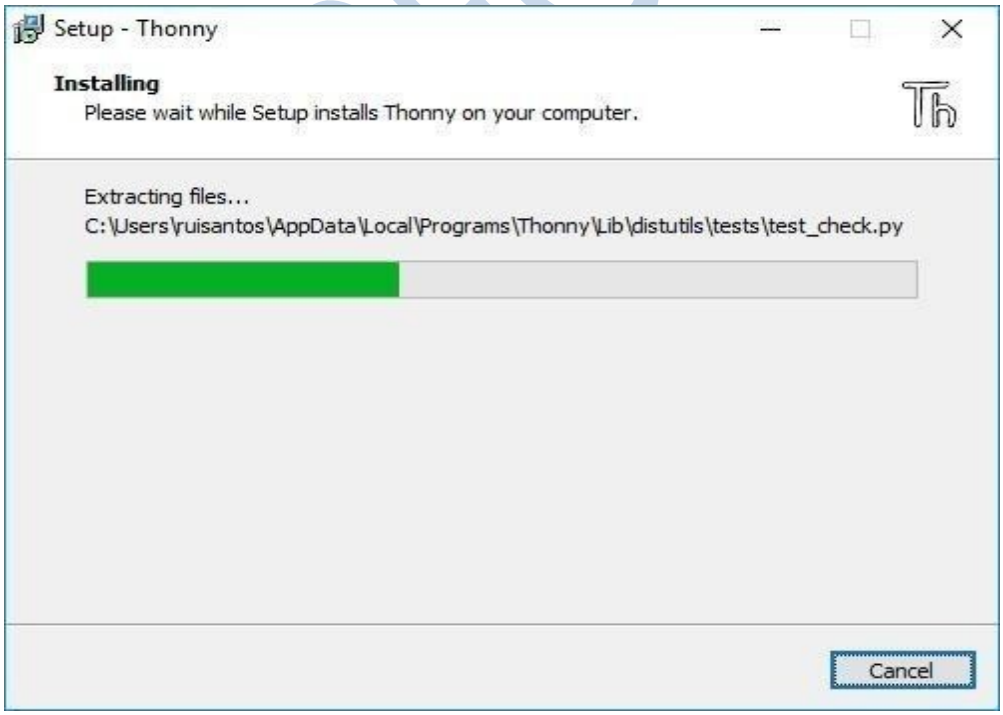
Thonny IDE comes installed by default on Raspbian OS that is used with the Raspberry Pi board. Toinstall Thonny on your Windows PC, follow the next instructions:

1. Go to https://thonny.org
2. Download the version for Windows and wait a few seconds while it downloads.

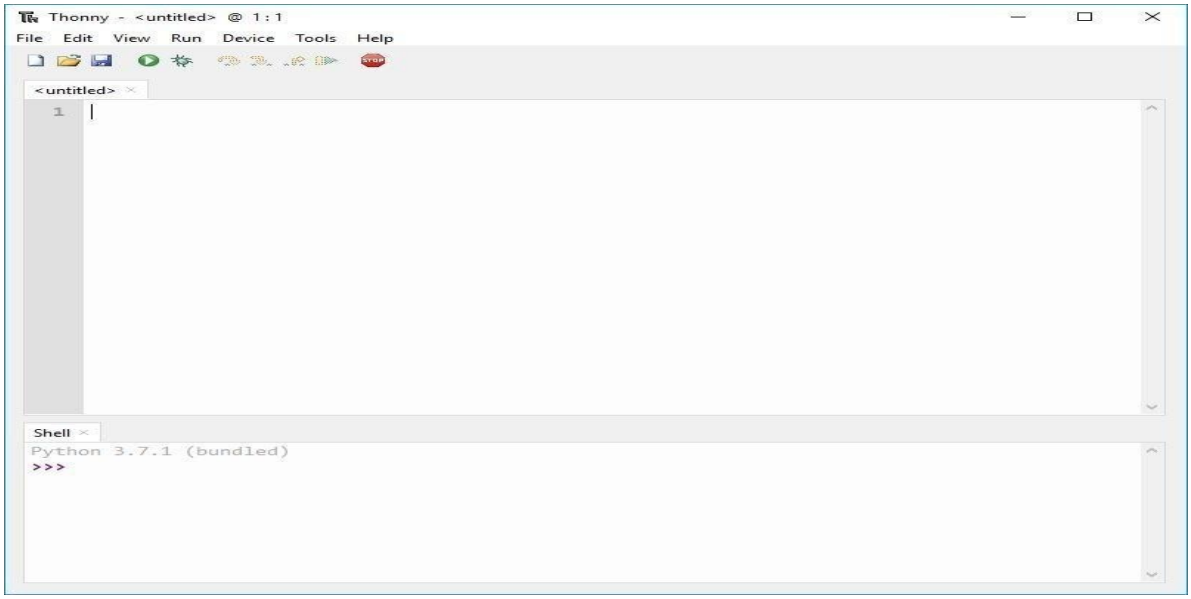




Follow the installation wizard to complete the installation process. You just need to click "Next".



After completing the installation, open Thonny IDE. A window as shown in the following figure should open.

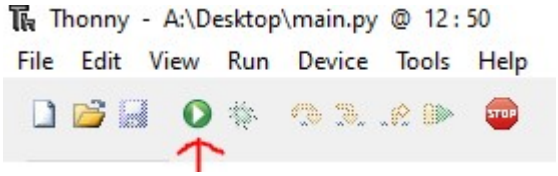


PROCEDURE:

- Connect your RP2040 Pico board to your computer using the USB cable.
- Open the Thonny IDE on your computer by clicking on 

.
- Go to "Tools" -> "Interpreter" -> "Details" and select "Try to detect automatically"
- Type the given program in the Thonny IDE.
- Save the program by clicking 

.
- Click "Stop" to stop any previously running Pico programs 

.
- Run the program by clicking play button
  - 

PROGRAM:

```
from machine import Pin
import time
led2 =Pin(2,Pin.OUT)
led3 =Pin(3,Pin.OUT)
while True:
    led2.toggle()
    led3.toggle()
    time.sleep(1)
```

CONNECTIONS:

Raspberry Pi Pico Pin	Raspberry Pi Pico Development Board
GP16	**LED**

APEC

RESULT:

Thus, the study of Python programming has been done and ensured its composition with internal features specifically.

Ex. No:14	INTERFACING SENSORS WITH RASPBERRY PI

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Thonny IDE	1
2	Raspberry Pi Pico Development Board	1
3	Jumper Wires	few
4	Micro USB Cable	1
5	IR Sensor	1
6	DHT sensor	1
7	LDR Sensor	1
8	LM35 Sensor	1

APEC

PROGRAM:

IR SENSOR

```
from machine import Pin
import time
ir_pin = Pin(0, Pin.IN)
while True:
    print("============================")
    ir_data=ir_pin.value()
    print("IR_SENSOR: ",ir_data)
    time.sleep(0.5)
```

DHT 11 SENSOR

```
from machine import Pin, I2C
import utime as time
from dht import DHT11, InvalidChecksum
while True:
    try:
        time.sleep(5)
        pin = Pin(28, Pin.OUT, Pin.PULL_DOWN)
        sensor = DHT11(pin)
        sensor.measure()
        t = (sensor.temperature)
        h = (sensor.humidity)
        print("Temperature: {}".format(sensor.temperature))
        print("Humidity: {}".format(sensor.humidity))
    except:
        print("error")
```

LDR SENSOR

```
from machine import Pin
import time
import utime
conversion_factor = 3.3/(65536)
adc2 = machine.ADC(28)
while True:
    val2 = adc2.read_u16()
    ldr = (val2 * conversion_factor)
    print("==================================")
    print("LDR Voltage : ",ldr)
```

APEC

```
    time.sleep(0.8)
```

LM35 SENSOR

```
from machine import Pin
import time
import utime
conversion_factor = 3.3/(65536)
adc2 = machine.ADC(27)

while True:
    val2 = adc2.read_u16()
    temp = (val2 * conversion_factor)*100.0
    temp2 = str(round(temp,2))
    print("================================")
    print("Temperature: ",temp2+" °C")
    time.sleep(0.8)
```

APEC

RESULT:

Interfacing the IR sensor, LDR, LM35 and DHT 11 sensor with Raspberry Pico has been verified andexecuted successfully

Ex. No:15	COMMUNICATE BETWEEN ARDUINO AND RASPBERRY PI

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Thonny IDE	1
2	Raspberry Pi Pico Development Board	1
3	Arduino Uno Development Board	1
4	Jumper Wires	few
5	Micro USB Cable	1
6	Bluetooth Module	2

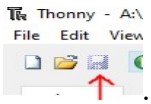
PROCEDURE:

1. Please refer to the "Firebase" document provided separately. It will tell you how to create and control your firebase account.
2. Here Arduino Uno acts as receiver and Raspberry Pico acts as transmitter.
3. Make the connections of HC-05 with Arduino Uno as given in the diagram above.
4. Make the connections of HC-05 with Raspberry Pico as given in the diagram above.
5. Connect the Arduino Uno board to your computer.
6. Open the Arduino IDE by clicking on 

.
7. Go to "Tools" -> "Port" and select the port to which Uno board is connected.
8. Again go to "Tools" -> "Board" and select Arduino Uno board.
9. Copy the program given below into the IDE and click "Upload".
10. Connect your RP2040 Pico board to your computer using the USB cable.
11. Open the Thonny IDE on your computer by clicking on 

.

APEC

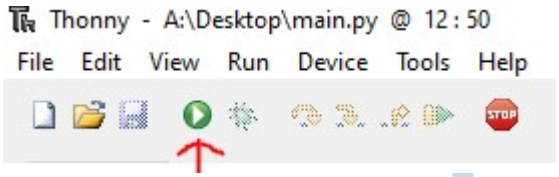
12. Go to "Tools" -> "Interpreter" -> "Details" and select "Try to detect automatically"
13. Type the given program in the Thonny IDE.
14. Save the program by clicking 

.
15. Connect your potentiometer to GP28 pin number.
16. Click "Stop" to stop any previously running Pico programs



.

17. Run the program by clicking play button

a. 

18. You will observe that Raspberry Pico sends "Hello" message and it is displayed in the serial monitor of Uno.

PROGRAM:

Code(Arduino Uno receiver side):

```
#include<SoftwareSerial.h>
SoftwareSerial bt(2,3); /* (Rx,Tx) */
void setup(){
 bt.begin(9600);
 Serial.begin(9600);
 Serial.println("=====HC05 Data Mode=====");
}
void loop(){
   if (bt.available())
   {
    Serial.write(bt.read());
   }
   if (Serial.available())
```

APEC

```
  {
    bt.write(Serial.read());
  }
}
```

Code(Raspberry Pico transmitter side):

```
import os
import machine
from time import sleep
uart = machine.UART(0, 9600)
print(uart)
b = None
while True:
  sleep(0.5)
  print("==========")
  uart.write("hello\n")
```

APEC

RESULT:

Thus, the program to Communicate between Arduino and Raspberry PI using any wireless medium (Bluetooth) has been verified and executed successfully.

Ex. No:16	CLOUD PLATFORM TO LOG THE DATA

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No.	Software Requirements	Quantity
1	Blynk Platform	1

CLOUD PLATFORM-BLYNK:



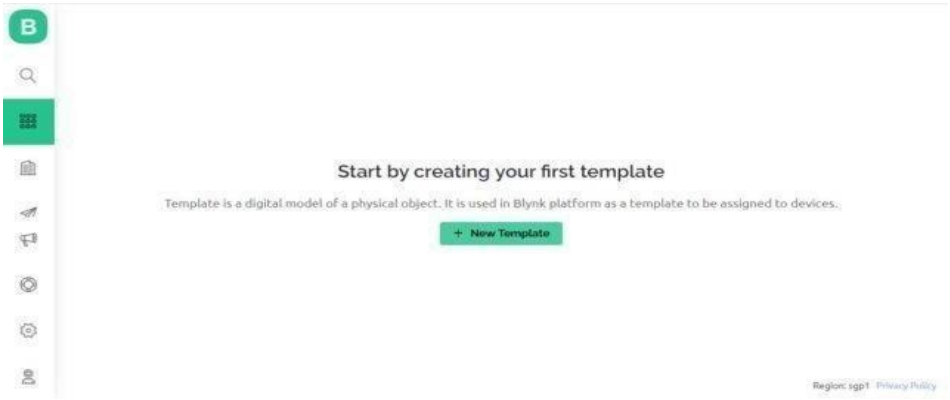
Blynk is a smart platform that allows users to create their Internet of Things applications without the need for coding or electronics knowledge. It is based on the idea of physical programming & provides aplatform to create and control devices where users can connect physical devices to the Internet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

Setting up Blynk 2.0 Application

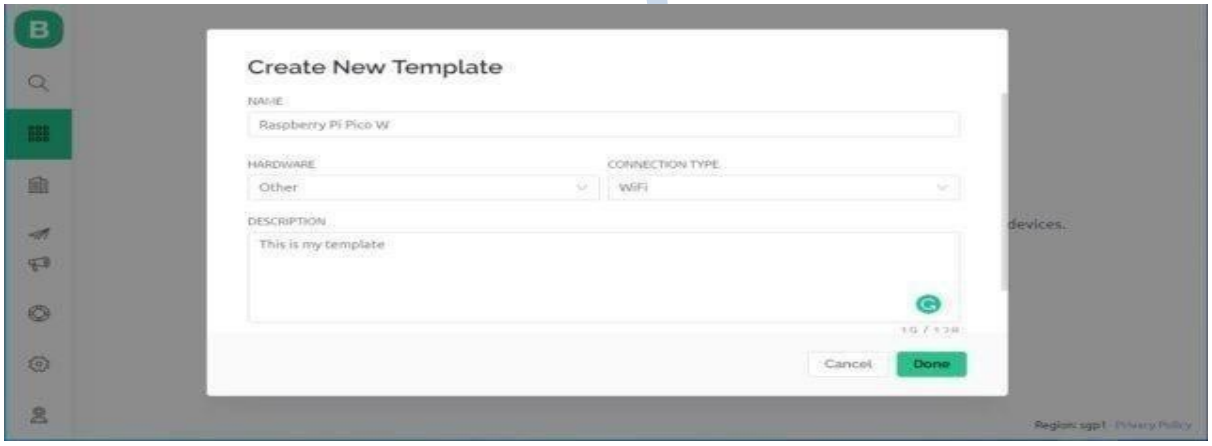
To control the LED using Blynk and Raspberry Pi Pico W, you need to create a Blynk project andset up a dashboard in the mobile or web application. Here's how you can set up the dashboard:

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

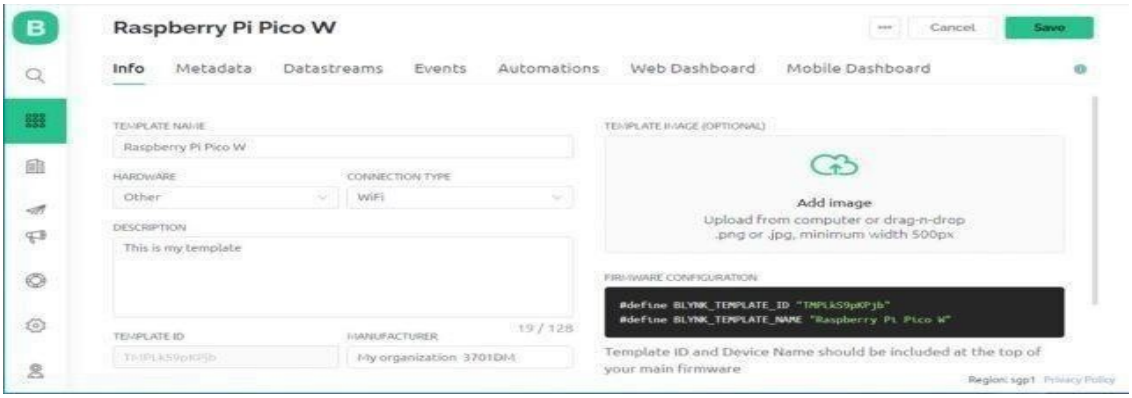
Step 2: Click on **+New Template**.



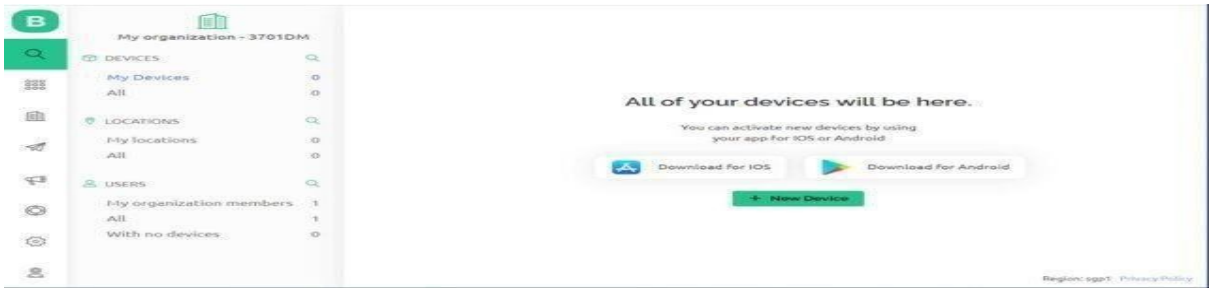
Step 3: Give any name to the Template such as Raspberry Pi Pico W. Select 'Hardware Type' asOther and 'Connection Type' as WiFi.



So a template will be created now.



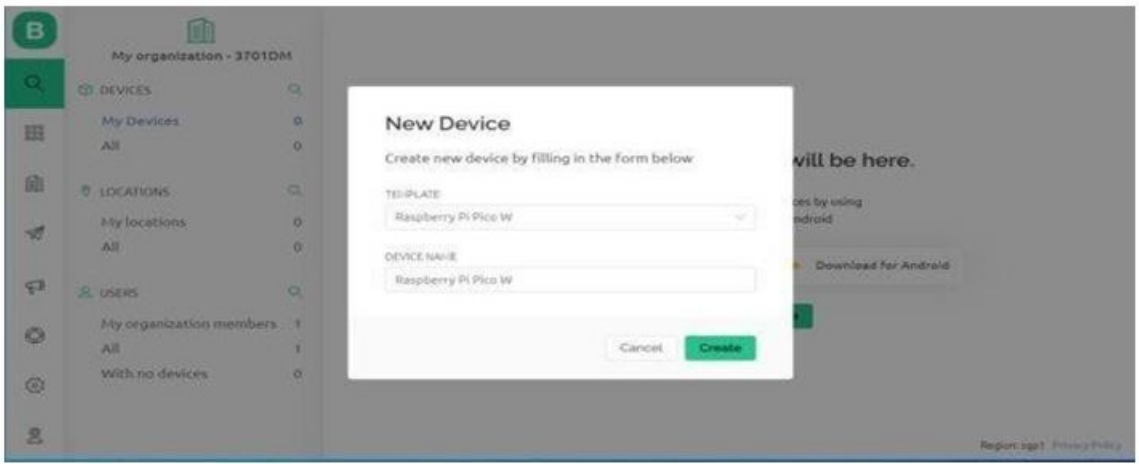
Step 4: Now we need to add a 'New Device' now.



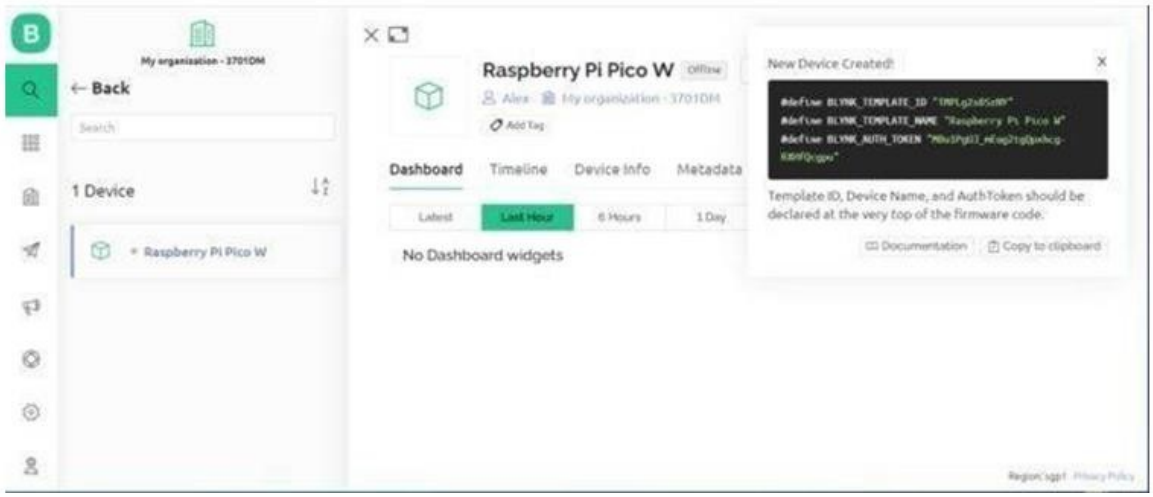
Select a New Device from ‘Template’



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



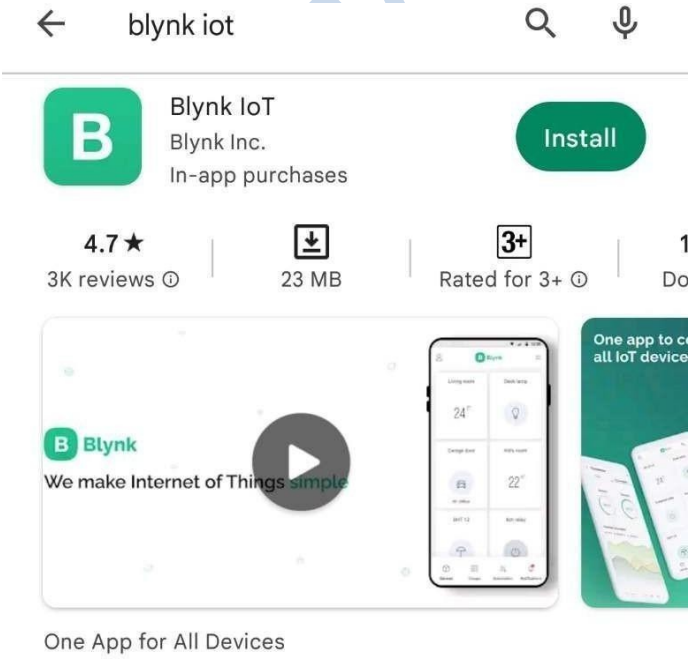
A new device will be created. You will find the Blynk Authentication Token Here. Copy it as it isnecessary for the code.



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

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

Step 7: To control the LED with a mobile App or Mobile Dashboard, you also need to setup the Mobile PhoneDashboard. 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 wecreated 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

APEC

RESULT:

Thus, a cloud platform to log the data from IoT devices has been verified and executedsuccessfully

Ex. No:17	LOG DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM

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	Jumper Wires	1
4	Micro USB Cable	1

PROGRAM:

```
from machine import Pin, I2C,
ADCfrom utime
import sleep_ms from pico_i2c_lcd
import I2cLcdimport 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'

# connect the networkwait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:break
    wait -= 1
    print('waiting for connection...')time.sleep(1)

# Handle connection errorif wlan.status() != 3:
    raise RuntimeError('network connection failed')else:
    print('connected') ip=wlan.ifconfig()[0]print('IP: ', ip)
```

APEC

```
  "Connection to Blynk"# Initialize Blynk
  blynk = BlynkLib.Blynk(BLYNK_AUTH)

lcd.clear() 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)
```

APEC

RESULT:

The program for Log Data using Raspberry PI and upload it to the cloud platform has been verified and executed successfully.

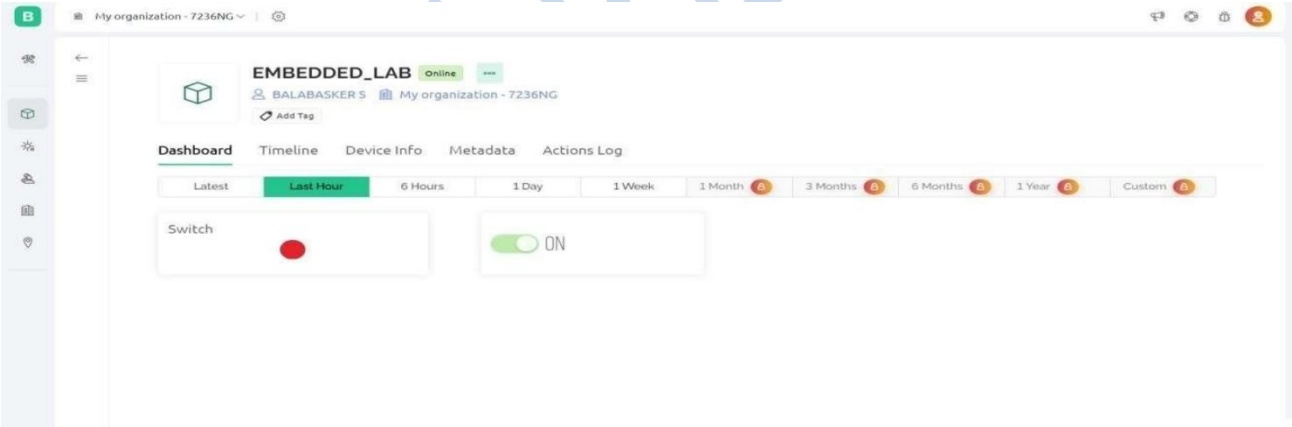
Ex. No:18	DESIGN AN IOT-BASED SYSTEM

AIM:

To design a Smart Home Automation IOT-based system

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	Thonny IDE	1
2	Raspberry Pi Pico Development Board	few
3	Jumper Wires	1
4	Micro USB Cable	1
5	LED or Relay	1
6	Home automation kit	1



CONNECTIONS:

Raspberry Pi Pico Pin	Raspberry Pi Pico Development Board
GP16	LED 1

PROGRAM:

```
#Initialize the library modules
import time
import network
from machine import Pin
import BlynkLib
from gpio_lcd import GpioLcd

#  Initialize the relay and lcd pins
relay1_pin = Pin(19, Pin.OUT,value=1)
relay2_pin = Pin(18, Pin.OUT,value=1)
relay3_pin = Pin(17, Pin.OUT,value=1)
relay4_pin = Pin(16, Pin.OUT,value=1)
lcd = GpioLcd(rs_pin = Pin(8),
        enable_pin = Pin(9),
        d4_pin = Pin(10),
        d5_pin = Pin(11),
        d6_pin = Pin(12),
        d7_pin = Pin(13))

# blynk Authendication token for connecting blynk cloud
BLYNK_AUTH ="xpGg_oOumLEgeWkyB0HNrFsGTJ_2l4VU"

#connecting to wifi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect("vivo","12345678")

# Wait for network connection
wait = 10
```

APEC

```
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print('waiting for connection...')
    lcd.move_to(0,0)
    lcd.putstr("waiting...")
    time.sleep(1)

# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
else:
    print('connected')
    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr("connected")
    ip = wlan.ifconfig()[0]
    print('IP: ', ip)

# Connect to Blynk
blynk = BlynkLib.Blynk(BLYNK_AUTH)

# print the text in LCD
lcd.clear()
lcd.move_to(0,0)
lcd.putstr("Light:")
v=0
for i in range(7,14,2):
    v=v+1
```

APEC

```
    lcd.move_to(i,0)
    lcd.putstr(str(v))
    time.sleep(0.2)
lcd.move_to(0,1)
lcd.putstr("State:")
for i in range(7,14,2):
    lcd.move_to(i,1)
    lcd.putstr("0")
    time.sleep(0.2)

# Register virtual pin handler
@blynk.on("V1") #virtual pin V1
def v1_write_handler(value): #read the value
    if int(value[0]) == 0:
        relay1_pin.value(1) #turn the relay1 on
        lcd.move_to(7,1)
        lcd.putstr("0")
    else:
        relay1_pin.value(0) #turn the relay1 off
        lcd.move_to(7,1)
        lcd.putstr("1")

@blynk.on("V2") #virtual pin V2
def v2_write_handler(value): #read the value
    if int(value[0]) == 0:
        relay2_pin.value(1) #turn the relay2 on
        lcd.move_to(9,1)
        lcd.putstr("0")
    else:
        relay2_pin.value(0) #turn the relay2 off
```

APEC

```
        lcd.move_to(9,1)
        lcd.putstr("1")

@blynk.on("V3") #virtual pin V3
def v3_write_handler(value): #read the value
    if int(value[0]) == 0:
        relay3_pin.value(1) #turn the relay3 on
        lcd.move_to(11,1)
        lcd.putstr("0")
    else:
        relay3_pin.value(0) #turn the relay3 off
        lcd.move_to(11,1)
        lcd.putstr("1")

@blynk.on("V4") #virtual pin V4
def v4_write_handler(value): #read the value
    if int(value[0]) == 0:
        relay4_pin.value(1) #turn the relay4 on
        lcd.move_to(13,1)
        lcd.putstr("0")
    else:
        relay4_pin.value(0) #turn the relay4 off
        lcd.move_to(13,1)
        lcd.putstr("1")
```

APEC

```
# main loop
while True:
    try:
        blynk.run()
        time.sleep(2)
    except:
        pass
```

APEC

RESULT:

The design of an IOT system has been verified and modelled successfully.