KARPAGA VINAYAGA

COLLEGE OF ENGINEERING AND TECHNOLOGY

Chinna Kolambakkam, Madhurantakam Tk, 603308

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CS3691

EMB EDED SYSTEM AND IOT LABORATORY

NAME :_____

REGISTER NO : _____

BRANCH :_____

SEM/YEAR :_____

INDEX

SNO.	DATE	EXPERIMENT TITLE	MARKS/10	SIGN.
1.		Addition of two 8-bit numbers		
2.		Subtraction of two 8-bit numbers		
3.		Multiplications of two 8-bit numbers		
4.		Division of two 8-bit numbers		
5.		Test data transfer between registers and memory		
6.		Programs using embedded c		
7.		Programs using Embedded C		
8.		Introduction to the Arduino platform		
9.		Programs using arduino uno		
10.		Programs using arduino uno for led blinking		
11.		Programs using arduino uno for bluetooth Communication		
12.		ZIGBEE Communication		
13.		Introduction to the Raspberry Pi Platform		
14.		Introduction to python programming		
15.		Interfacing sensors with raspberry pi		
16.		Communicate between Arduino and Raspberry pi		
17.		Cloud platform to log the data		
18.		Log data using raspberry pi and upload it tothe cloud platform		

Program: 1	**Addition of two 8-bit numbers**.
DATE:	

AIM:

To write and execute an Assembly language program to perform addition of two 8-bit numbers.

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 forthe 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:

```
ORG 00H
SJMP START
ORG  30H
START:
CLR  C
MOV A, #20H
MOV B, #21H
ADD A, B
MOV R0, A
END
```

INPUT:

A=20h
B=21h

OUTPUT:

R0=41h

RESULT:

Thus, the program for the addition of two 8-bit numbers is executed and the results are verified successfully using Keil Software.

Program: **2**	**Subtraction of two 8-bit numbers**
DATE:	

AIM:

To write and execute an Assembly language program to perform Subtraction of two 8-bit numbers.

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 forthe 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:

```
ORG 00H
        SJMP    START
        ORG     30H
        START:

CLR C
MOV A, #25H
MOV B, #04H
SUBB A, B
MOV R0, A
MOV R1, B
END
```

INPUT:

A=25h
B=04h

OUTPUT:

R0=21h
R1=00h

RESULT:

Thus, the program for the subtraction of two 8-bit numbers is executed and the results are verified successfully using Keil Software.

Program: 3

DATE:	Multiplications of two 8-bit numbers

AIM:

To write and execute an Assembly language program to perform Multiplications of two 8-bit numbers.

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 forthe 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:

```
ORG 00H
SJMP START
ORG 30H
START:

CLR C              //MULTIPLICATION
MOV A, #03H
MOV B, #04H
MUL AB
MOV R0, A
MOV R1, B
END
```

INPUT:

A=03h
B=04h

OUTPUT:

R0=0Ch
R1=00h

RESULT:

Thus, the program for the multiplication of two 8-bit numbers is executed and the results are verified successfully using Keil Software.

Program: 4	**Division of two 8-bit numbers**
DATE:	

AIM:

To write and execute an Assembly language program to perform Division of two 8-bit numbers.

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 forthe 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:

```
ORG 00H
        SJMP    START
        ORG     30H
        START:


CLR C       //DIVISION
 MOV A, #45H
MOV B, #04H
DIV AB
MOV R0, A
MOV R1, B
END
```

INPUT:

A=45h
B=04h

OUTPUT:

R0=
R1=

RESULT:

Thus, the program for the multiplication of two 8-bit numbers is executed and the results are verified successfully using Keil Software.

Program: 5	**Test data transfer between registers and memory**
DATE:	

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:

```
ORG 0000H
CLR C
MOV  R0,  #30H
MOV  R1,  #40H
MOV R2, #05H
BACK: MOV A, @R0
MOV @R1, A
INC R0
INC R1
DJNZ R2, BACK
END
```

INPUT DATA:

S.NO	Source Memory Location	Data's	Destination Memory Location	Data's
1	30	11	40	11
2	31	22	41	22
3	33	33	42	33
4	34	44	43	44
5	35	55	44	55

RESULT:

Thus, the execution of program to transfer the data between memories is done successfully.

Program: 6	**PROGRAMS USING EMBEDDED C**
DATE:	

AIM:

Write program to load three numbers into Accumulator and sending them to port1 using Embedded C Language.

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 forTarget.
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 .c 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
8. Step run.

PROGRAM:

```
#include <reg51.h>
void main ()
{
ACC=0x25;
P1=ACC;
ACC=0x46;
P1=ACC;
ACC=0x92;
P1=ACC;
}
```

INPUT DATA and OUTDATA:

A=25h

0	**0**	**1**	**0**	**0**	**1**	**0**	**1**

A=46h

0	**1**	**0**	**0**	**0**	**1**	**1**	**0**

A=92h

1	**0**	**0**	**1**	**0**	**0**	**1**	**0**

RESULTS:

Thus, the program to load three numbers into Accumulator and sending them to PORT1 is done successfully.

Program: 7	**PROGRAMS USING EMBEDDED C**
DATE:	

AIM:

To write an embedded C program **to send values 00 – FF to port P1.** using the Keil software.

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 .c 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>
 void main ()
 {
  unsigned char z;
  for (z = 0; z <= 255; z++)
  P1=Z;
 }
```

INPUT DATA:

Z= 0 to 255

OUTPUT DATA at PORT 1Z=00H

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Z=01H

0	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Z=02H

0	**0**	**0**	**0**	**0**	**0**	**2**	**0**

.

.

.

Z=FFH

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

RESULT:

Thus, the embedded C program to send values 00 – FF to port P1 using the Keil softwareis executed and verified the values successfully.

EXP NO: 08 DATE	INTRODUCTION TO THE ARDUINO PLATFORM

AIM:

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

Hardware & SOFTWARE TOOLS REQUIRED:

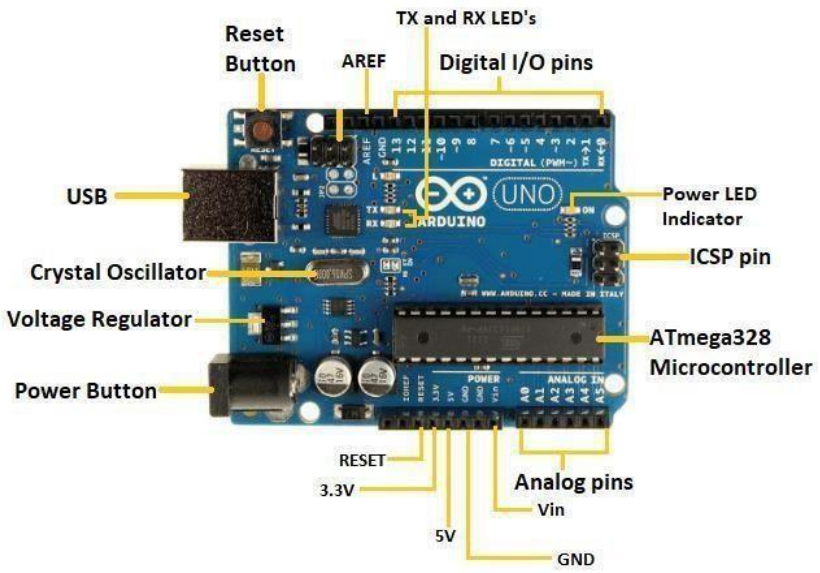
S.No.	Hardware & Software Requirements	Quantity
1	Arduino IDE 2.0	1
2	Arduino Uno Board	1

INTRODUCTION TO ARDUINO:

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.

The Arduino board is shown below:

The components of Arduino UNO board are shown below:



Let's discuss each component in detail.

- **ATmega328 Microcontroller**- It is a single-chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines **Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timers, external and internal interrupts, and oscillator.**
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The ON status of the LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LED's**- The successful flow of data is represented by the lighting of these LED's.
- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino

  UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pin.

TECHNICAL SPECIFICATIONS OF ARDUINO UNO

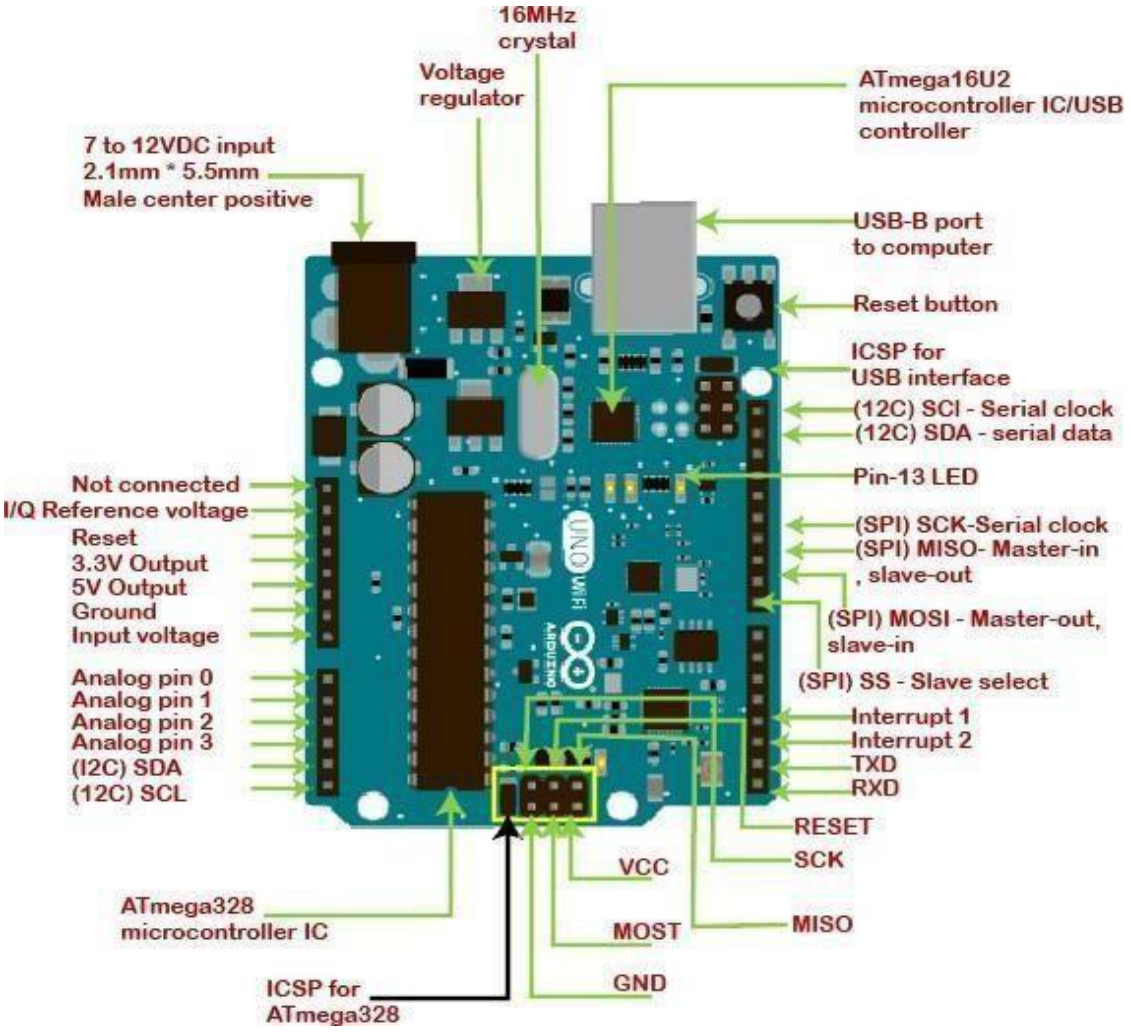
The technical specifications of the Arduino UNO are listed below:

- There are 20 Input/Output pins present on the Arduino UNO board. These 20 pins include 6 PWM pins, 6 analog pins, and 8 digital I/O pins.
- The PWM pins are Pulse Width Modulation capable.
- The crystal oscillator present in Arduino UNO comes with a frequency of 16MHz.
- It also has an Arduino-integrated WIFI module. Such Arduino UNO board is based on the Integrated WIFI ESP8266 Module and ATmega328P microcontroller.
- The input voltage of the UNO board varies from 7V to 20V.
- Arduino UNO automatically draws power from the external power supply. It can also draw power from the USB.

ARDUINO UNO PINOUT

The Arduino UNO is a standard board of Arduino, which is based on an **ATmega328P** microcontroller. It is easier to use than other types of Arduino Boards.

The Arduino UNO Board, with the specification of pins, is shown below:



Let's discuss each pin in detail.

ATmega328 Microcontroller- It is a single chip Microcontroller of the ATmel family. The processor core inside it is of 8-bit. It is a low-cost, low powered, and a simple microcontroller. The Arduino UNO and Nano models are based on the ATmega328 Microcontroller.

Voltage Regulator: The voltage regulator converts the input voltage to 5V. The primary function of voltage regulator is to regulate the voltage level in the Arduino board. For any changes in the input voltage of the regulator, the output voltage is constant and steady.

GND - Ground pins. The ground pins are used to ground the circuit.

TXD and RXD: TXD and RXD pins are used for serial communication. The TXD is used for transmitting the data, and RXD is used for receiving the data. It also represents the successful flow of data.

USB Interface: The USB Interface is used to plug-in the USB cable. It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.

RESET: It is used to add a Reset button to the connection.

SCK: It stands for **Serial Clock**. These are the clock pulses, which are used to synchronize the transmission of data.

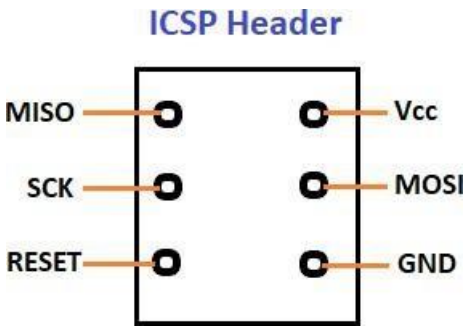
MISO: It stands for **Master Input/ Slave Output**. The save line in the MISO pin is used to send the data to the master.

VCC: It is the modulated DC supply voltage, which is used to regulate the IC's used in the connection. It is also called as the primary voltage for IC's present on the Arduino board. The Vcc voltage value can be negative or positive with respect to the GND pin.

Crystal Oscillator- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.

ICSP: It stands for **In-Circuit Serial Programming**. The users can program the Arduino board's firmware using the ICSP pins. The program or firmware with the advanced functionalities is received by microcontroller with the help of the ICSP header. The ICSP header consists of 6 pins.

The structure of the ICSP header is shown below:



SDA: It stands for **Serial Data**. It is a line used by the slave and master to send and receive data. It is called

as a **data line,** while SCL is called as a clock line.

SCL: It stands for **Serial Clock**. It is defined as the line that carries the clock data. It is used to synchronize the transfer of data between the two devices. The Serial Clock is generated by the device and it is called as master.

SPI: It stands for **Serial Peripheral Interface**. It is popularly used by the microcontrollers to communicate with one or more peripheral devices quickly. It uses conductors for data receiving, data sending, synchronization, and device selection (for communication).

MOSI: It stands for Master Output/ Slave Input. The MOSI and SCK are driven by the Master.

SS: It stands for **Slave Select**. It is the Slave Select line, which is used by the master. It acts as the enable line.

I^2C: It is the two-wire serial communication protocol. It stands for Inter Integrated Circuits. The I^2C is a serial communication protocol that uses SCL (Serial Clock) and SDA (Serial Data) to receive and send data between two devices.

3.3V and 5V are the operating voltages of the board.

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 it's 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

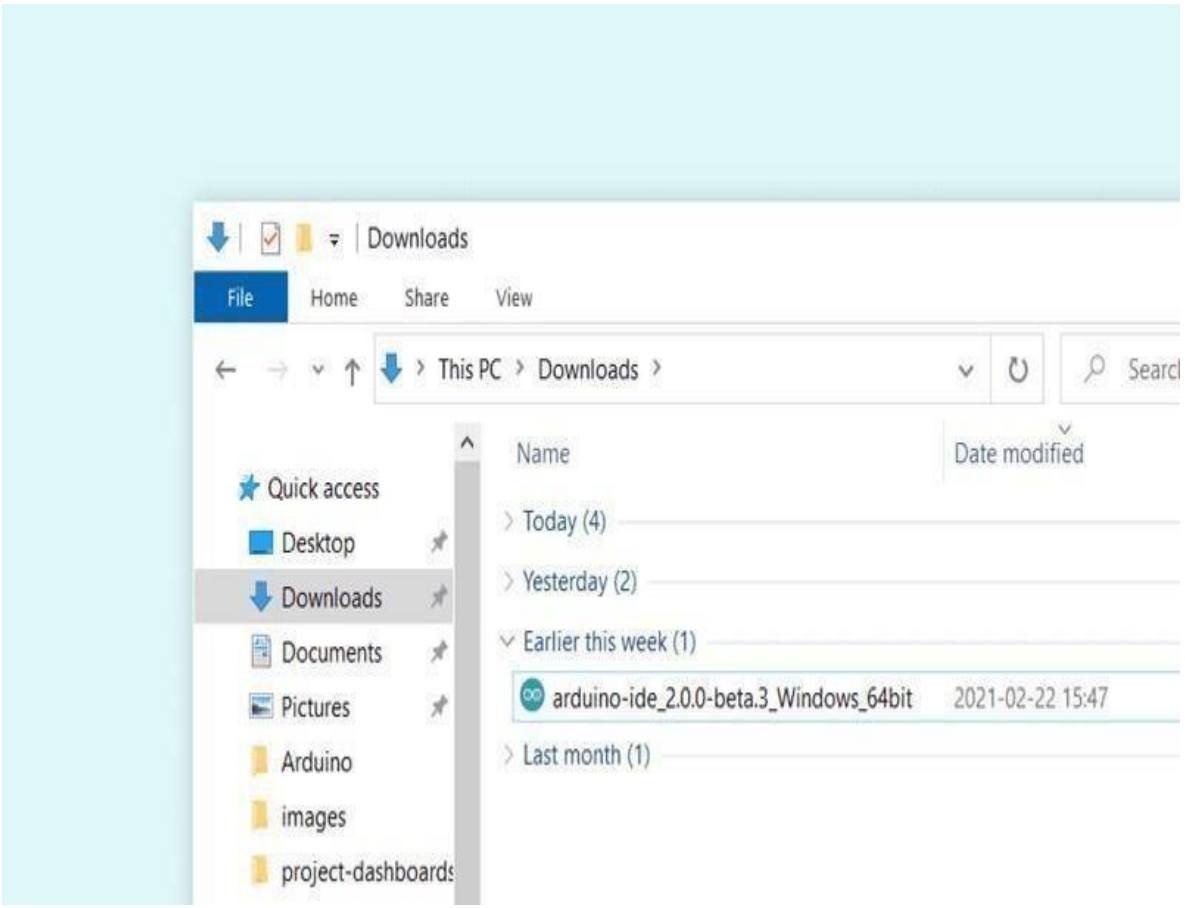
- **Windows** - Win 10 and newer, 64 bits
- **Linux** - 64 bits
- **Mac OS X** - Version 10.14: "Mojave" or newer, 64 bits

Installation

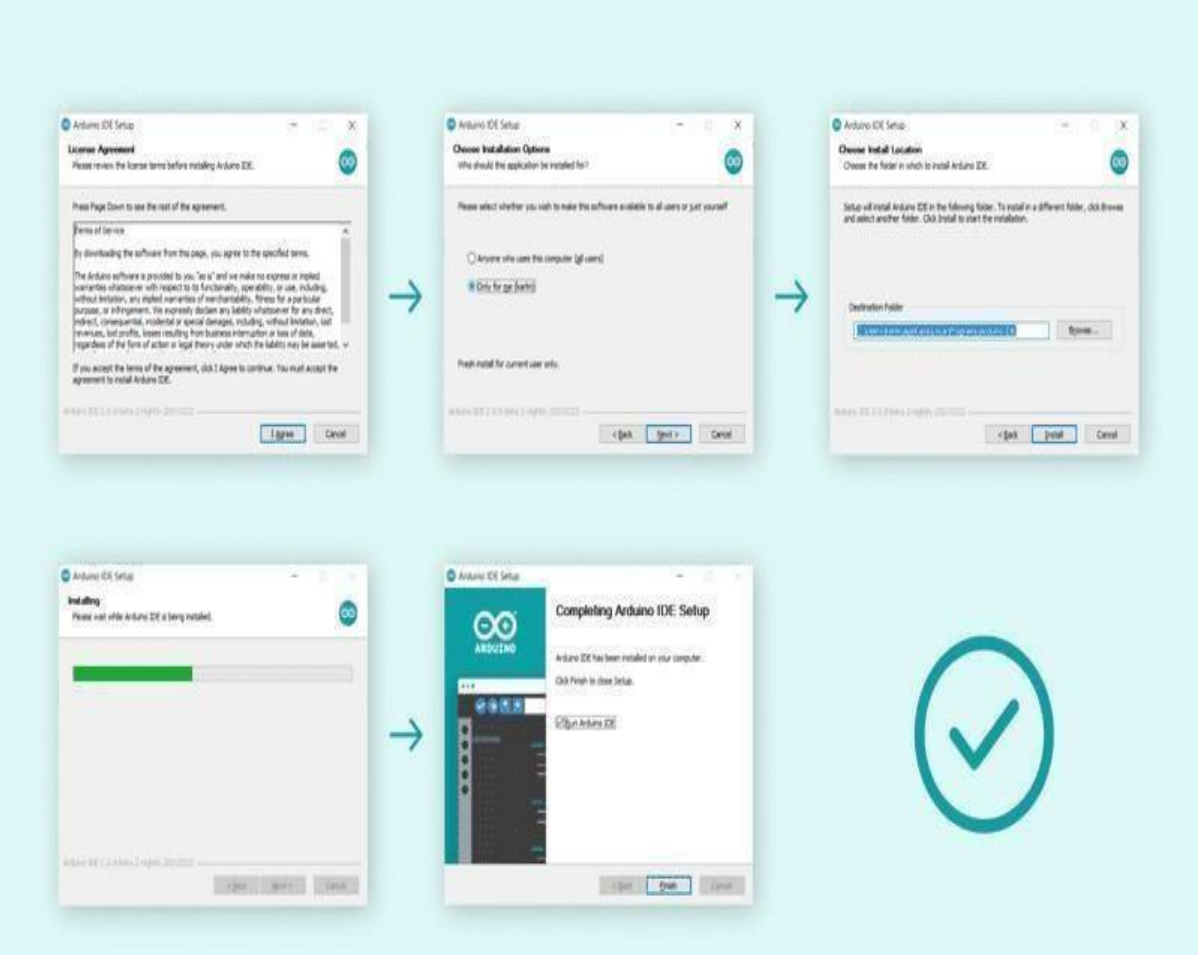
Windows

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.


Downloads
File
Home
Share
View
This PC
Downloads
Quick access
Desktop
Downloads
Documents
Pictures
Arduino
images
project-dashboards
Name
Date modified
Today (4)
Yesterday (2)
Earlier this week (1)
arduino-ide_2.0.0-beta.3_Windows_64bit
2021-02-22 15:47
Last month (1)

Follow the instructions in the installation guide. The installation may take several minutes.



You can now use the Arduino IDE 2.0 on your windows computer!

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.

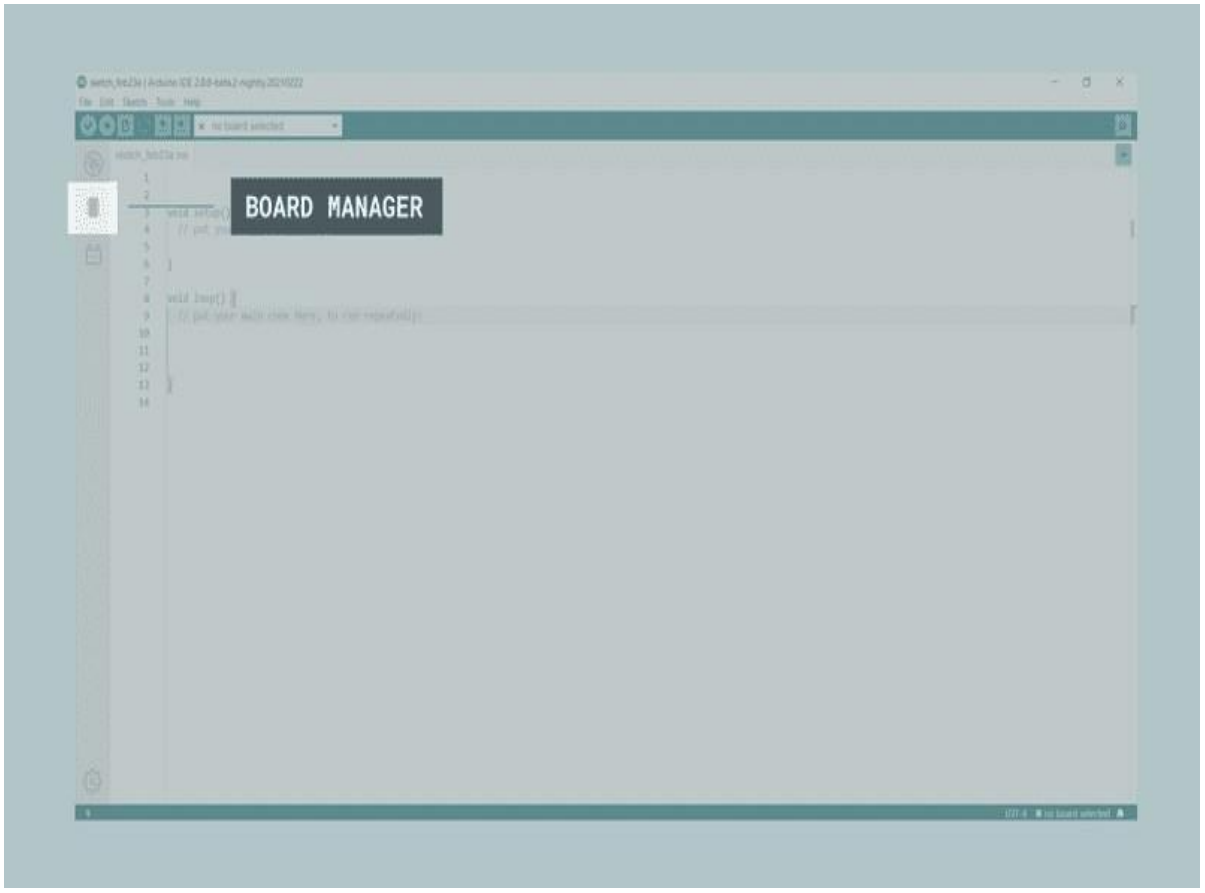
For example, an Arduino UNO has an **ATmega328P**, which uses the **AVR core**, while an Arduino Nano 33 IoT has a **SAMD21** microcontroller, where we need to use the **SAMD core**.

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

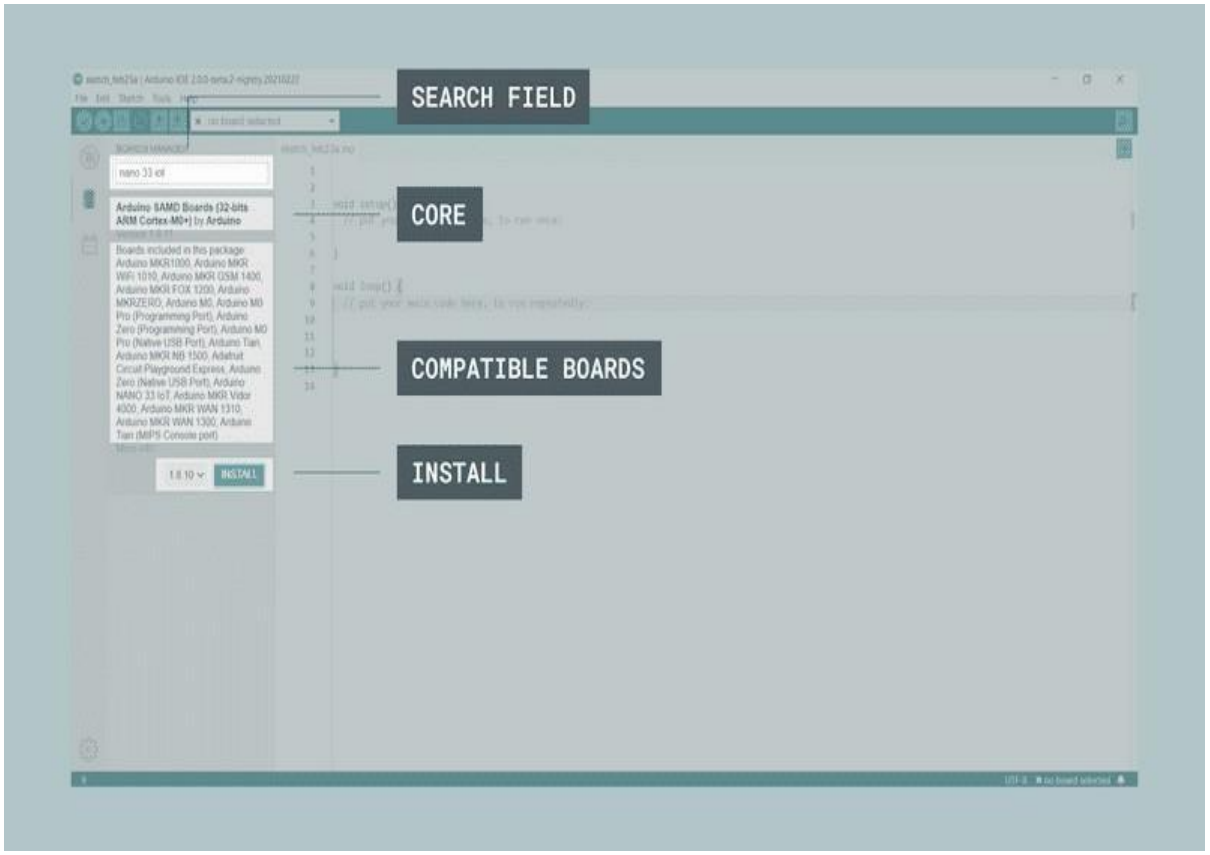
Installing a core

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

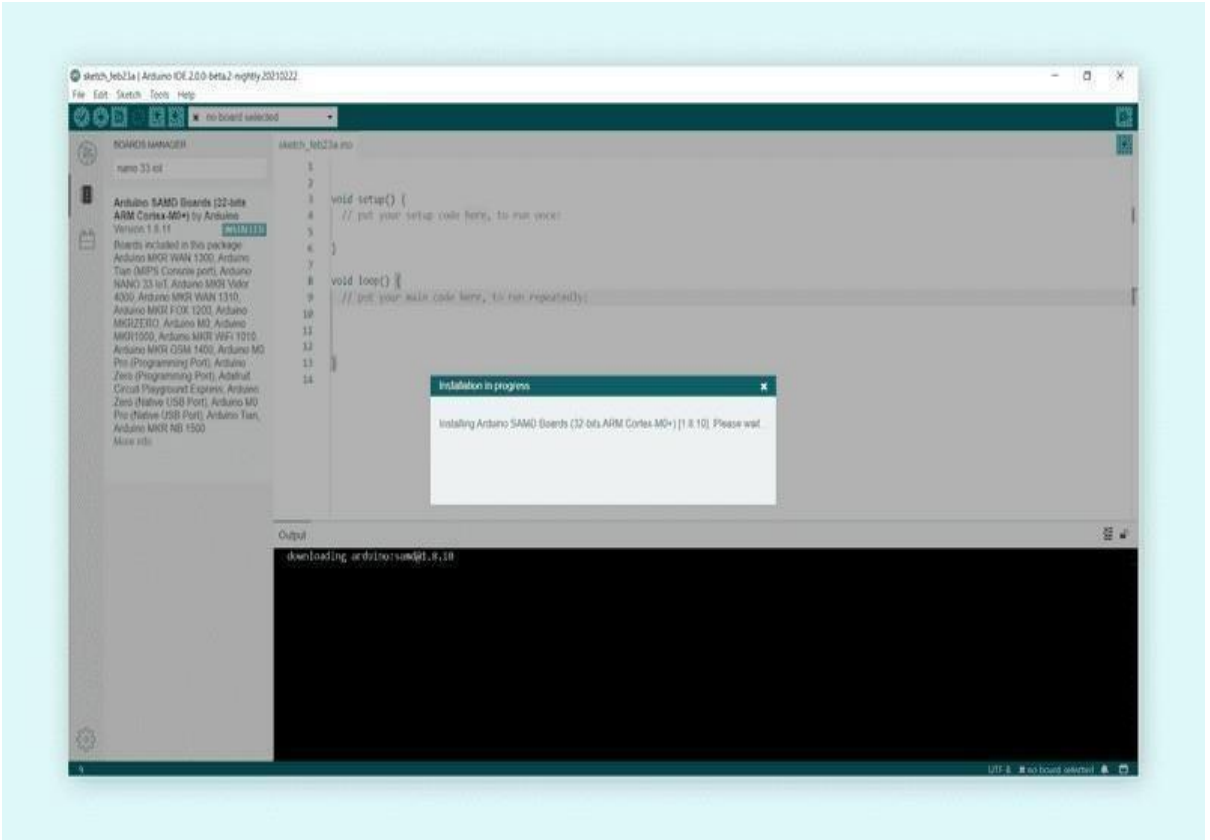
1. Open the Arduino IDE 2.0.
2. With the editor open, let's take a look at the left column. Here, we can see a couple of icons. Let's click the on the **"computer chip"** icon.



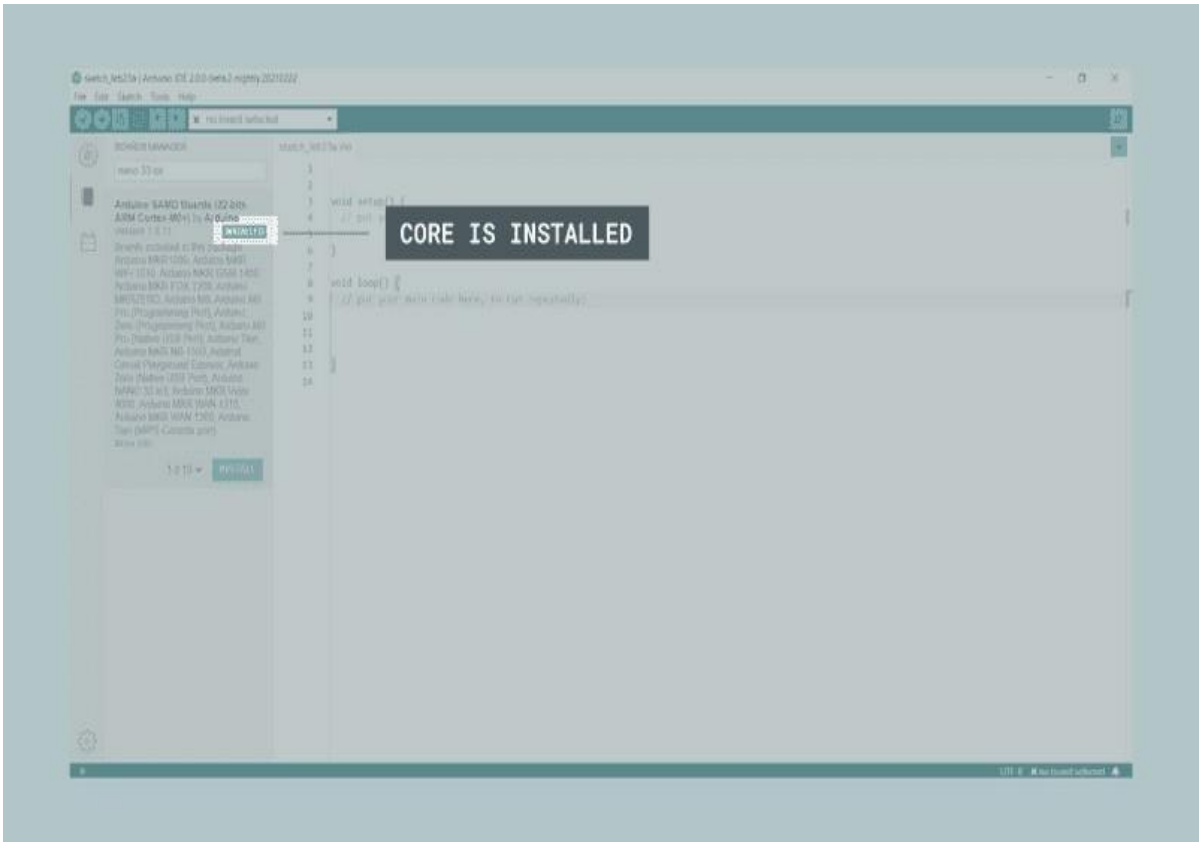
1. A list will now appear of all available cores. Now let's say we are using an **Nano 33 IoT** board, and we want to install the core. Simply enter the name in the search field, and the right core (SAMD) will appear, where the Nano 33 IoT features in the description. Click on the **"INSTALL"** button.



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



5. 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.

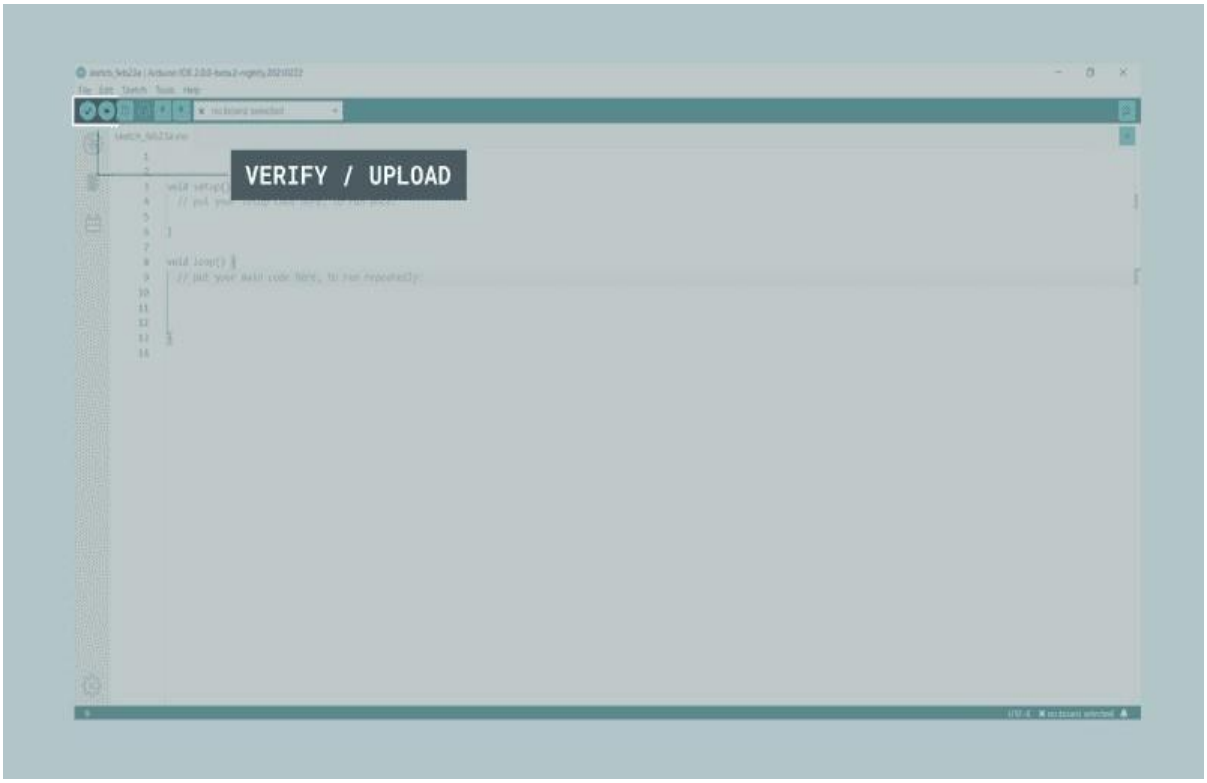
A good practice is to use the verifying tool before attempting to upload anything. This is a quick way of spotting any errors in your code, so you can fix them before actually uploading the code.

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

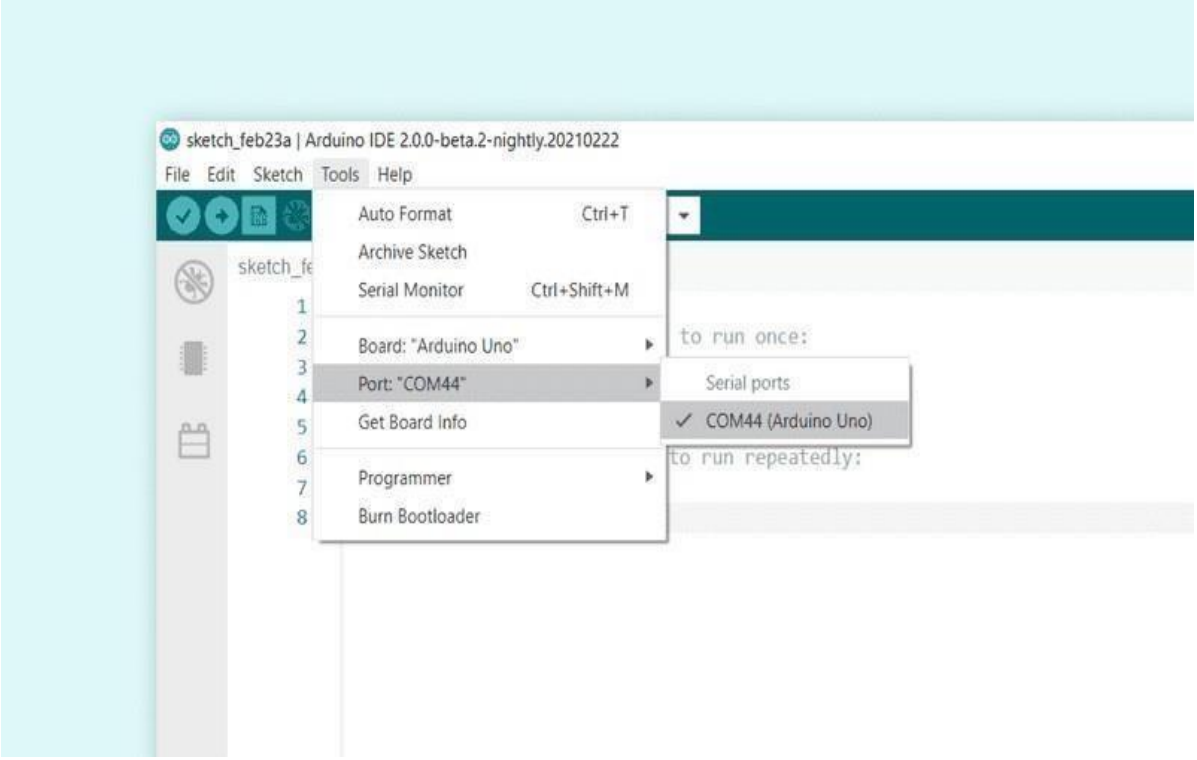
is a **checkmark** and an **arrow pointing right**. The checkmark is used to **verify**, and the arrow is used to **upload.**



3. 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).

1. 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)**.



5. With the board selected, we are good to go! Click on the **upload** button, and it will start uploading the sketch to the board.
6. 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.

you have now uploaded a sketch to your Arduino board!

How to install and use a library with the Arduino IDE 2.0

A large part of the Arduino programming experience is the **use of libraries.** Thousands of libraries can be found online, and the best-documented ones can be found and installed directly through the editor. In this tutorial, we will go through how to install a library using the library manager in the Arduino IDE 2.0. We will also show how to access examples from a library that you have installed.

Requirements

- Arduino IDE 2.0 installed.

Why use libraries?

Libraries are incredibly useful when creating a project of any type. They make our development experience much smoother, and there almost an infinite amount out there. They are used to interface with many different sensors, RTCs, Wi-Fi modules, RGB matrices and of course with other components on your board.

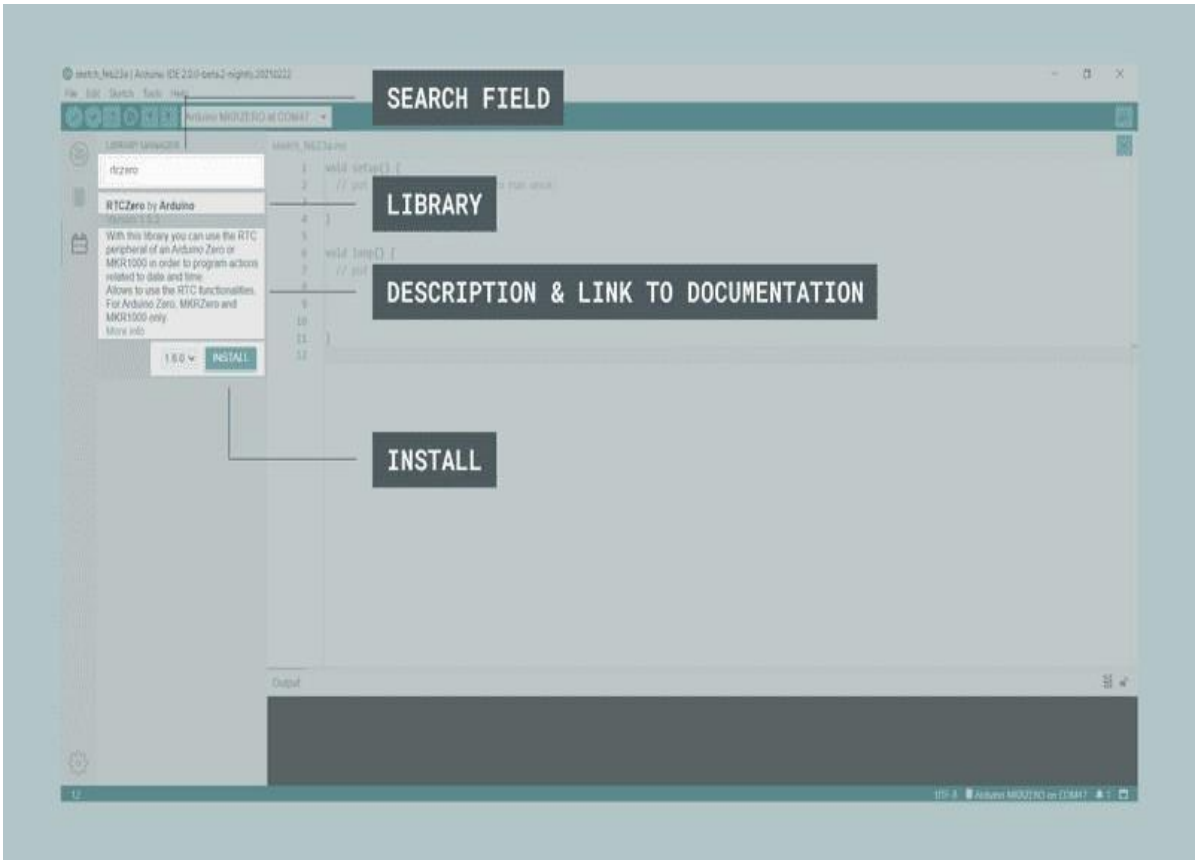
Arduino has many official libraries, but the real heroes are the Arduino community, who develop, maintain and improve their libraries on a regular basis.

Installing a library

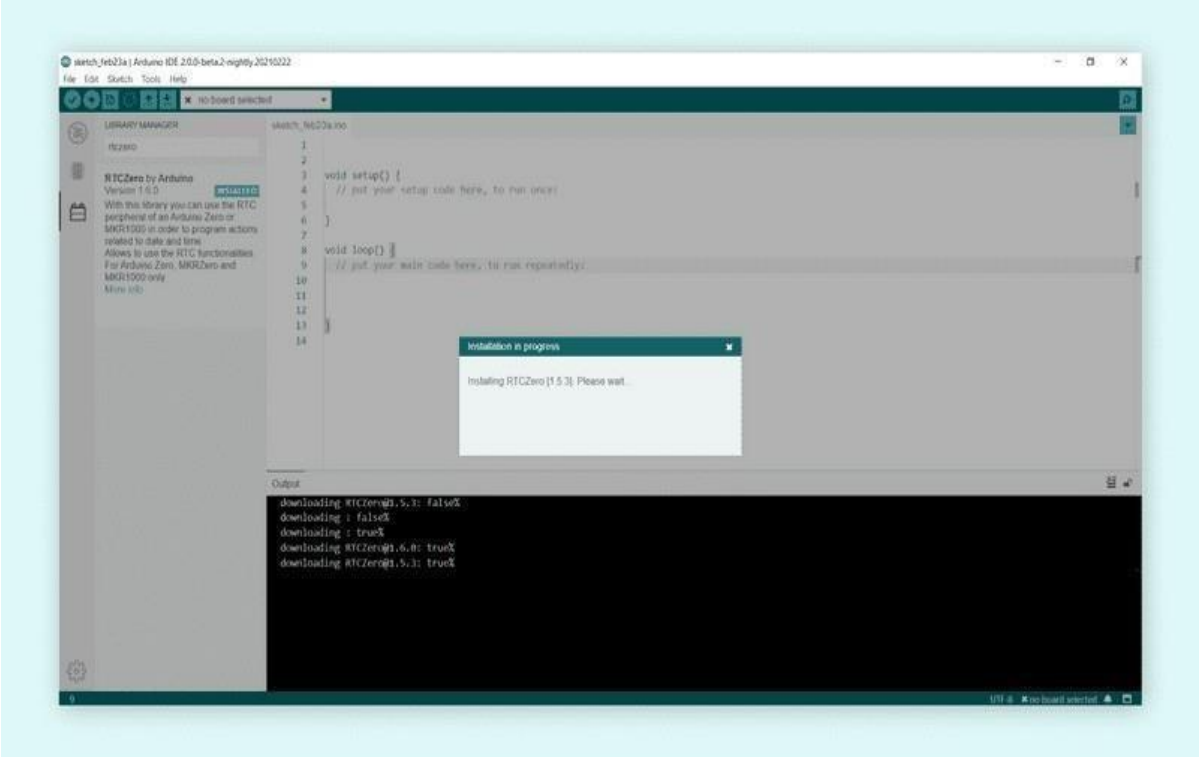
Installing a library is quick and easy, but let's take a look at what we need to do.

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 **"library"** icon.

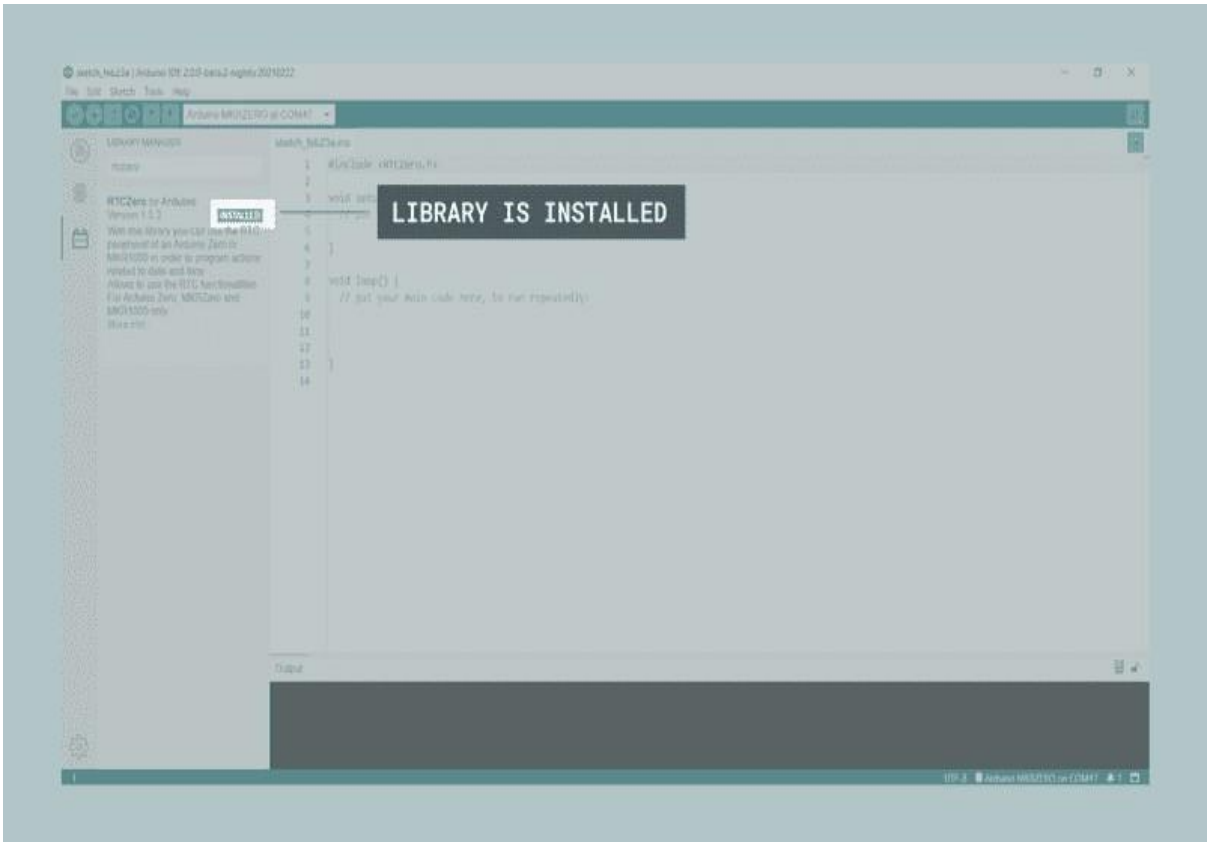
3. A list will now appear of all available libraries, where we can also search for the library we want to use. In this example, we are going to install the **RTCZero** library. Click on the **"INSTALL"** button to install the library.



4. This process should not take too long, but allow up to a minute to install it.



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



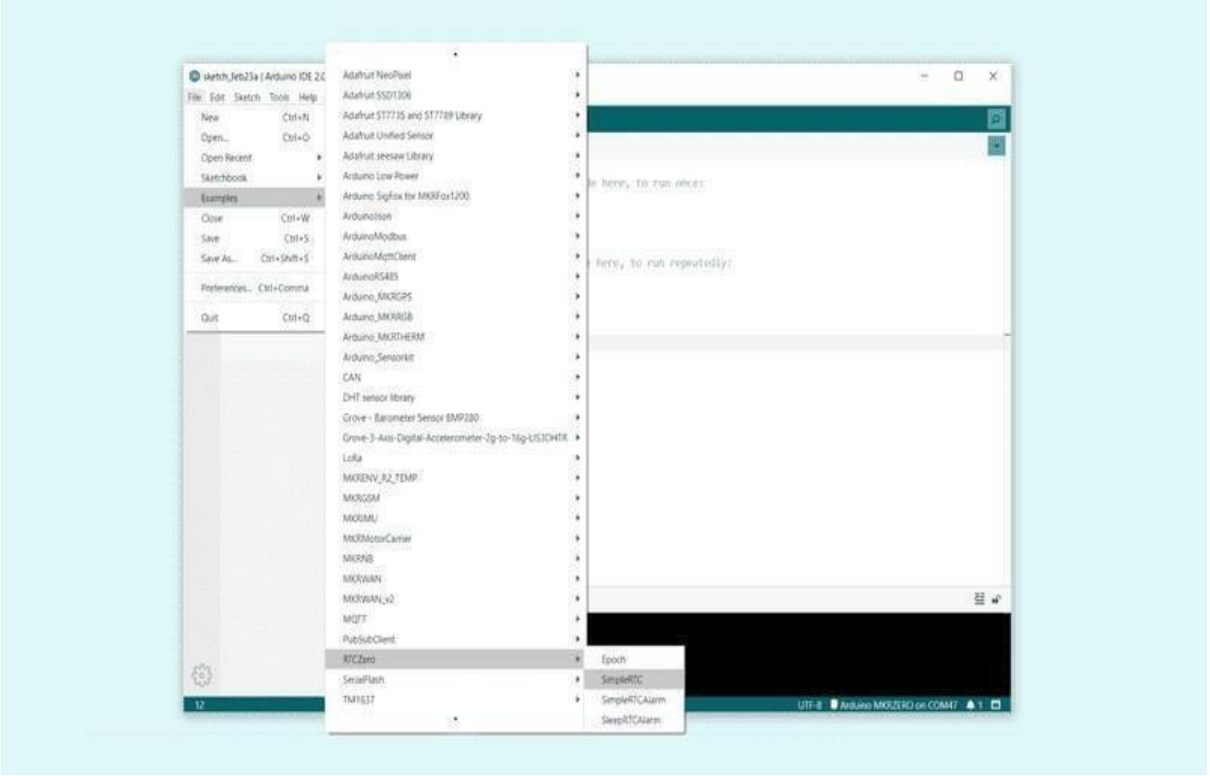
You have now successfully downloaded and installed a library on your machine.

Including a library

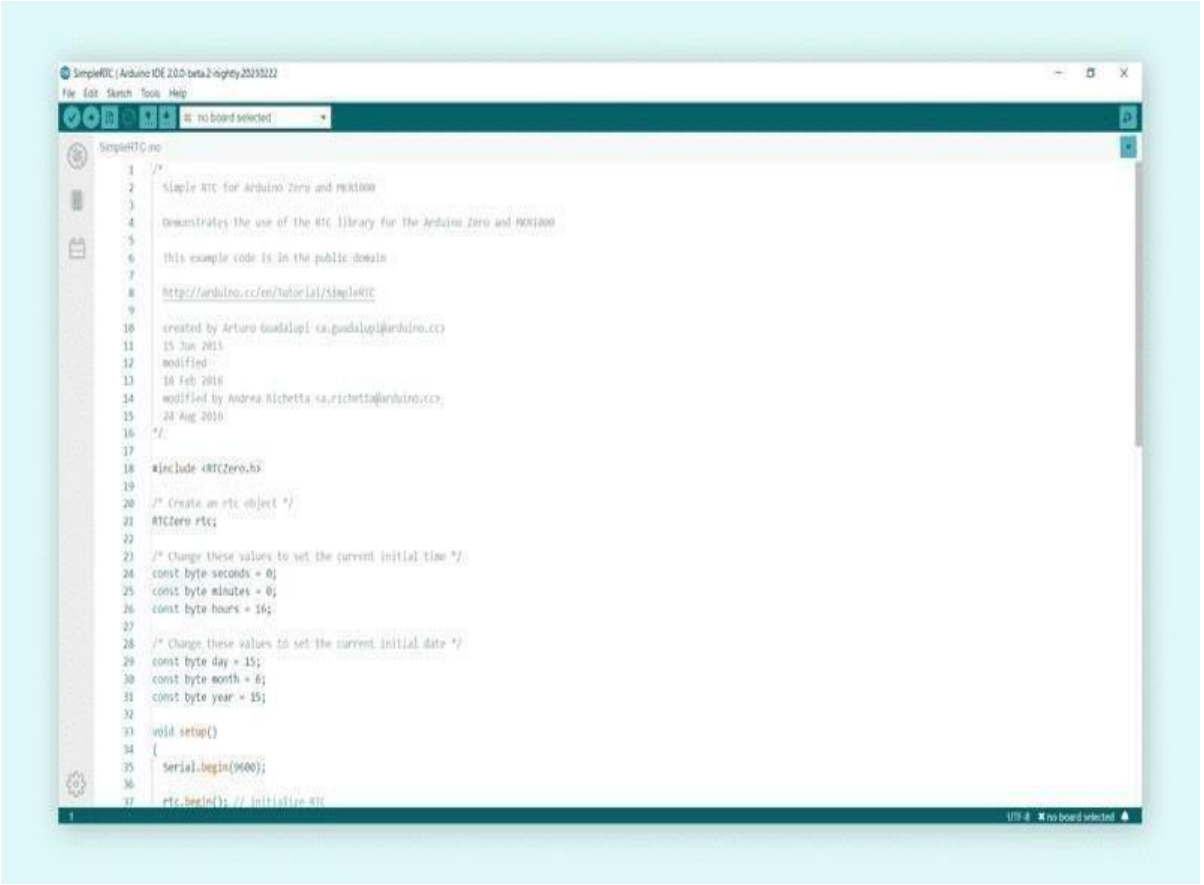
To use a library, you first need to include the library at the top of the sketch.

Almost all libraries come with already made examples that you can use. These are accessible through

File > Examples > {Library} > {Example}. In this example, we are choosingthe **RTCZero > SimpleRTC**.



The chosen example will now open up in a new window, and you can start using it however you want to.


SimpleRTC | Arduino IDE 2.0.0-beta.2-nightly.20210222
File Edit Sketch Tools Help
no board selected
SimpleRTC.ino
/*
Simple RTC for Arduino Zero and MKR1000
Demonstrates the use of the RTC library for the Arduino Zero and MKR1000
this example code is in the public domain
http://arduino.cc/en/Tutorial/SimpleRTC
created by Arturo Guadalupi <a.guadalupi@arduino.cc>
15 Jun 2015
modified
10 Feb 2016
modified by Andrea Richetta <a.richetta@arduino.cc>
24 Aug 2016
*/
#include <RTCZero.h>
/* Create an rtc object */
RTCZero rtc;
/* Change these values to set the current initial time */
const byte seconds = 0;
const byte minutes = 0;
const byte hours = 16;
/* Change these values to set the current initial date */
const byte day = 15;
const byte month = 6;
const byte year = 15;
void setup()
{
Serial.begin(9600);
rtc.begin(); // initialize RTC
UTF-8 no board selected

RESULT:

Program: 9	**PROGRAMS USING ARDUINO UNO**
DATE:	

AIM:

To write an Arduino program to making a sequential LED blinking circuit with Arduino Uno

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	ARDUINO UNO **IDE**	**1**

1. Arduino Uno r3
2. Data cable to connect Arduino with laptop or pc

3.3 LEDs (Color of your choice)

4.1 Breadboard

5.16 male to male jumper wires

6.A 5v power source

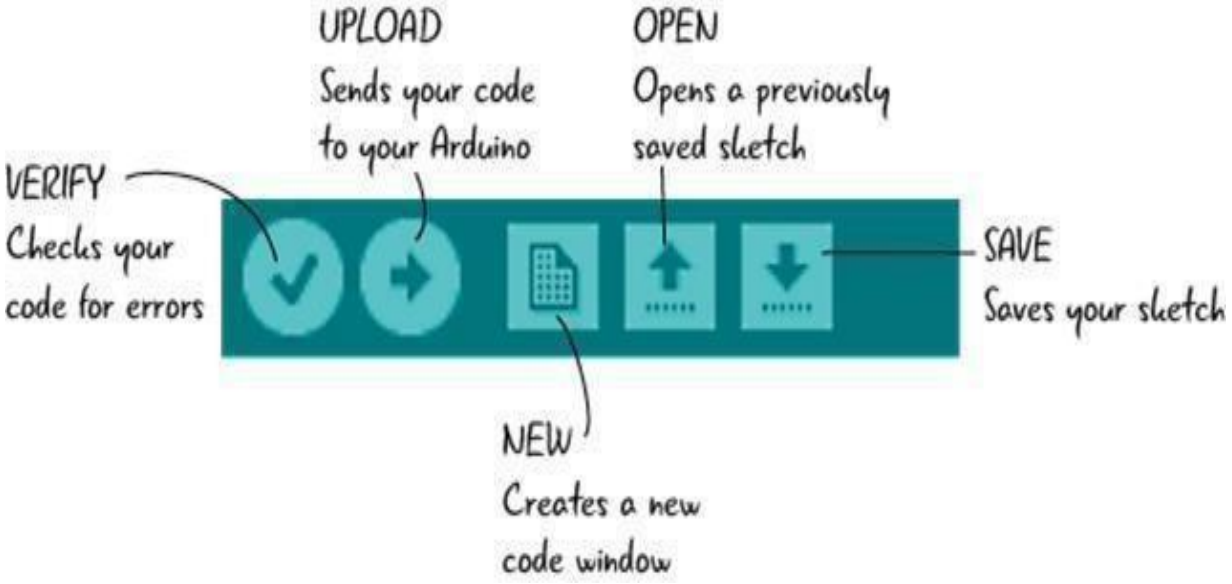
7.Arduino IDE Software

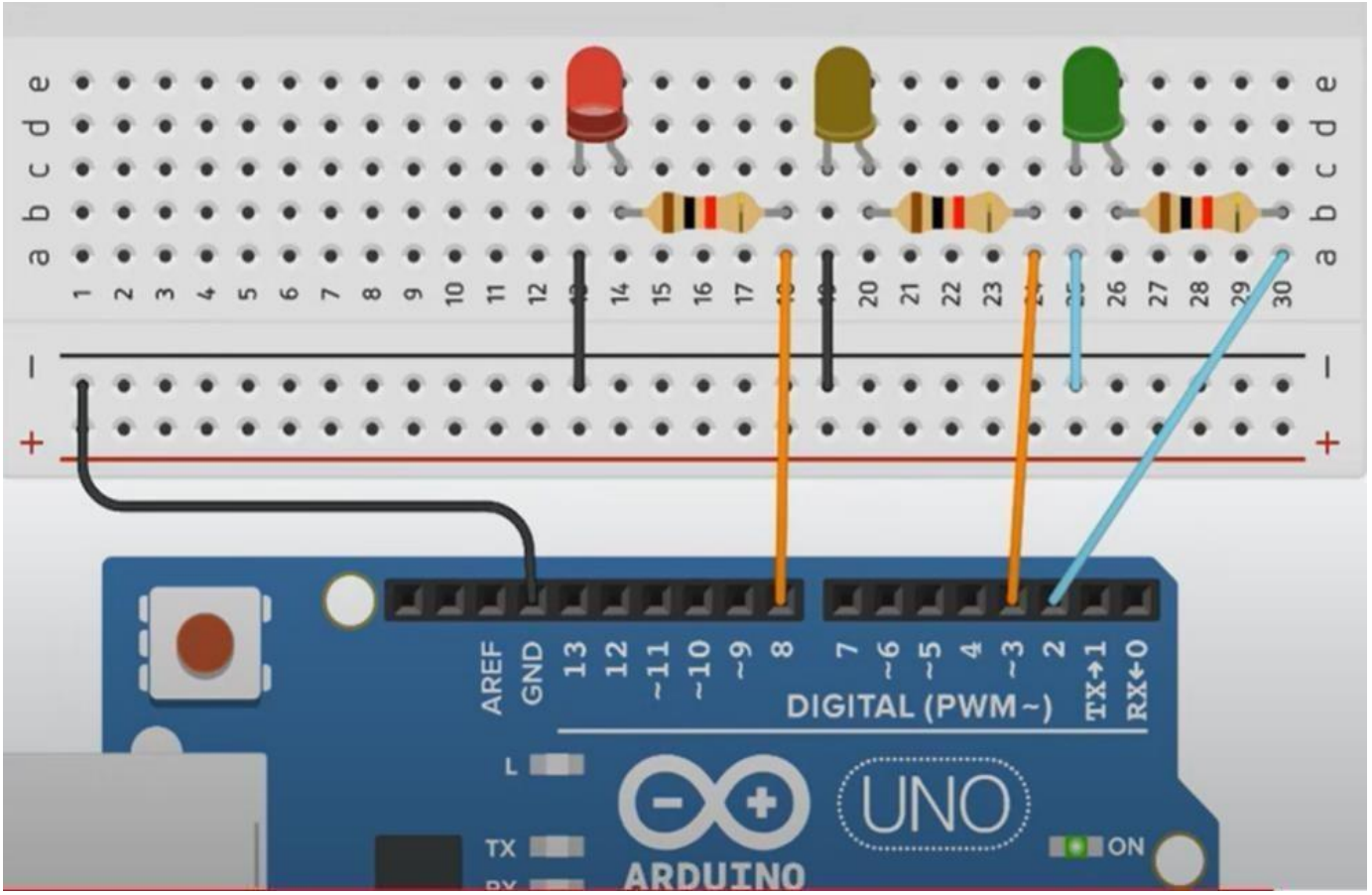
Procedure:

1. Connect the digital GND pin of the Arduino to any of the power rail on the breadboard.
2. Next, connect the LEDs to the breadboard with the Negative (-) legs towards the left as shown in the image below:
3. Now connect the Negative (-) legs of all the LEDs to the power rail on which you connected the GND pin of Arduino using jumper wires as shown in the image below
4. Now connect the 1st LED to pin no.8, 2nd LED to pin 3, 3rd LED to pin 2 using jumper wires as shown in the video below
5. Now open the Arduino IDE software on your laptop or pc.
6. Now, you can copy and paste the code under the specified columns:

PROGRAMS: This is the code: (above the void setup)

```
int LED1 = 8;

int LED2 = 3;

int LED3 = 2;

void setup ()

{
pinMode(LED1, OUTPUT);

pinMode(LED2, OUTPUT);

pinMode(LED3, OUTPUT);

}
void loop()
{
digitalWrite(LED1, HIGH);

delay(200);

digitalWrite(LED2, HIGH);

delay(200);

digitalWrite(LED3, HIGH);

delay(200);

digitalWrite(LED1, LOW);

delay(300);

digitalWrite(LED2, LOW);

delay(300);

digitalWrite(LED3, LOW);

delay(300);

}
```





RESULTS:

Thus, the Arduino program to making a sequential LED blinking circuit with Arduino Uno is executed and results are verified at LED's.

*

Program: 10	PROGRAMS USING ARDUINO UNO FOR LED BLINKING
DATE:	

AIM:

To write an Arduino program to making a LED blinking circuit with

Arduino Uno

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	ARDUINO UNO **IDE**	**1**

Components Required

1. 1 X LED
2. 1 X Resistor, 330 Ohm
3. Breadboard
4. Arduino UNO R4 or earlier versions.
5. Jumper wires

Components Description

1. 1 X LED: We are controlling only one LED in this program.
2. 1 X Resistor, 330 Ohm: For every LED, we need one current limiting resistor.
3. Breadboard: A breadboard is a fundamental tool used in electronics and prototyping to build and test circuits without soldering.
4. Arduino UNO R4 or earlier versions.
5. Jumper wires: Jumper wires are simple electrical wires with connectors on both ends used to create connections between various electronic components or points on a circuit on a breadboard.*

PROGRAMS:

```
int LEDpin = 13;
int delayT = 1000;

void setup()
{

// put your setup code here, to run once:

pinMode(LEDpin, OUTPUT);

}

void loop()

{                    // put your main code here, to run repeatedly:

digitalWrite(LEDpin, HIGH);

delay(delayT);

digitalWrite(LEDpin,  LOW);

delay(delayT);

}
```

RESULTS:

Thus, the Arduino program to making a sequential LED blinking circuit with Arduino Uno is executed and results are verified at LED's.

Program: 11	PROGRAMS USING ARDUINO UNO FORBLUETOOTH COMMUNICATION
DATE:	

AIM:

To write an Arduino program to making a light ON/OFF with Bluetooth module circuit with Arduino Uno

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	ARDUINO UNO **IDE**	**1**
2	Bluetooth App	

Components Required

1. Arduino UNO Board-1
2. Bluetooth module-1
3. Relay 10A-1
4. Bulb and switch
5. Jumper wires

Procedure:

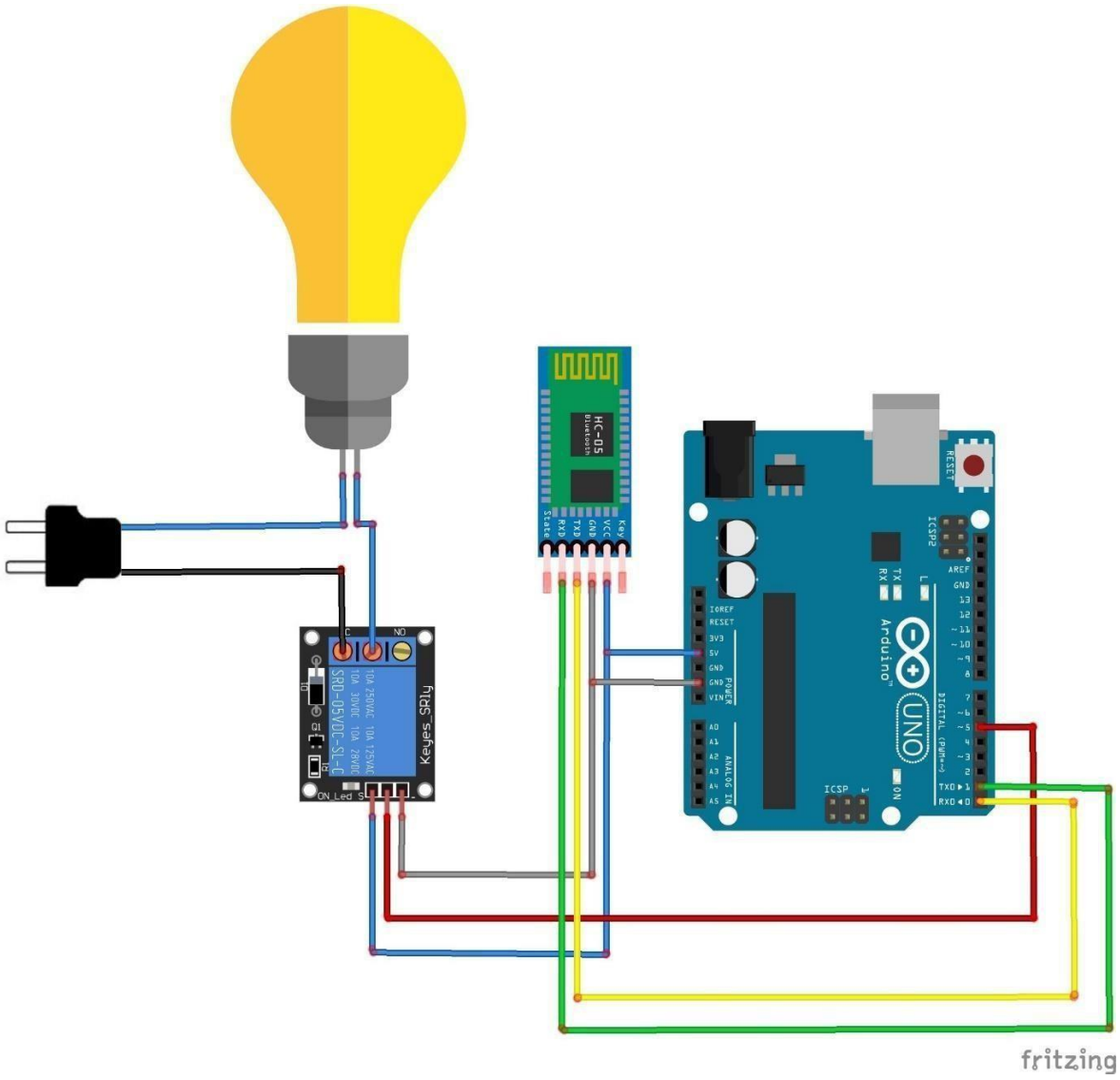


Figure: Arduino based home light ON/OFF ushing Bluetooth

PROGRAM: This is the code: (above the void setup)

```
char val;

void setup()

{

pinMode(5, OUTPUT);
Serial.begin(9600);
digitalWrite(5, HIGH);
}

void loop()

{

 if(Serial.available())
 {
 val = Serial.read();
 Serial.println(val);
}

if(val=='1')

{

digitalWrite(5, LOW);

}

else if(val=='3')

{

digitalWrite(5, HIGH);

}

delay(100);

}
```

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 idealfor 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.

RESULTS

EXP NO:12	ZIGBEE COMMUNICATION
DATE	

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

PROCEDURE

Follow the following instructions to configure the Zigbee series 1 module :

Sure, here's the corrected version:

1. Connect your Zigbee module to the computer's serial port using a serial adapter.
2. Download a virtual terminal program like PuTTY for Windows.
3. Configure your computer's serial port settings to a baud rate of 9,600, 8 data bits, no parity, and 1 stop bit.
4. Enable the "Local Echo" option.
5. Click "OK" to save the session settings.
6. Click the "Connect" button in the virtual terminal program.
7. Once connected to the Zigbee module, give your session a name.
8. Utilize AT commands to configure your module as needed.

After that, type +++ on the virtual terminal. After a few seconds, it will respond with "OK". Then, type these commands:

CONNECTIONS:

TRANSMITTER:

Arduino UNO Pin	Zigbee Module
VCC	5V
GND	G
2	Tx
3	Rx

CONNECTIONS:

RECEIVER:

Arduino UNO Pin	Zigbee Module	Arduino Development Board
-	5V	5V
-	G	GND
2	Tx	-
3	Rx	-
4	-	LED1

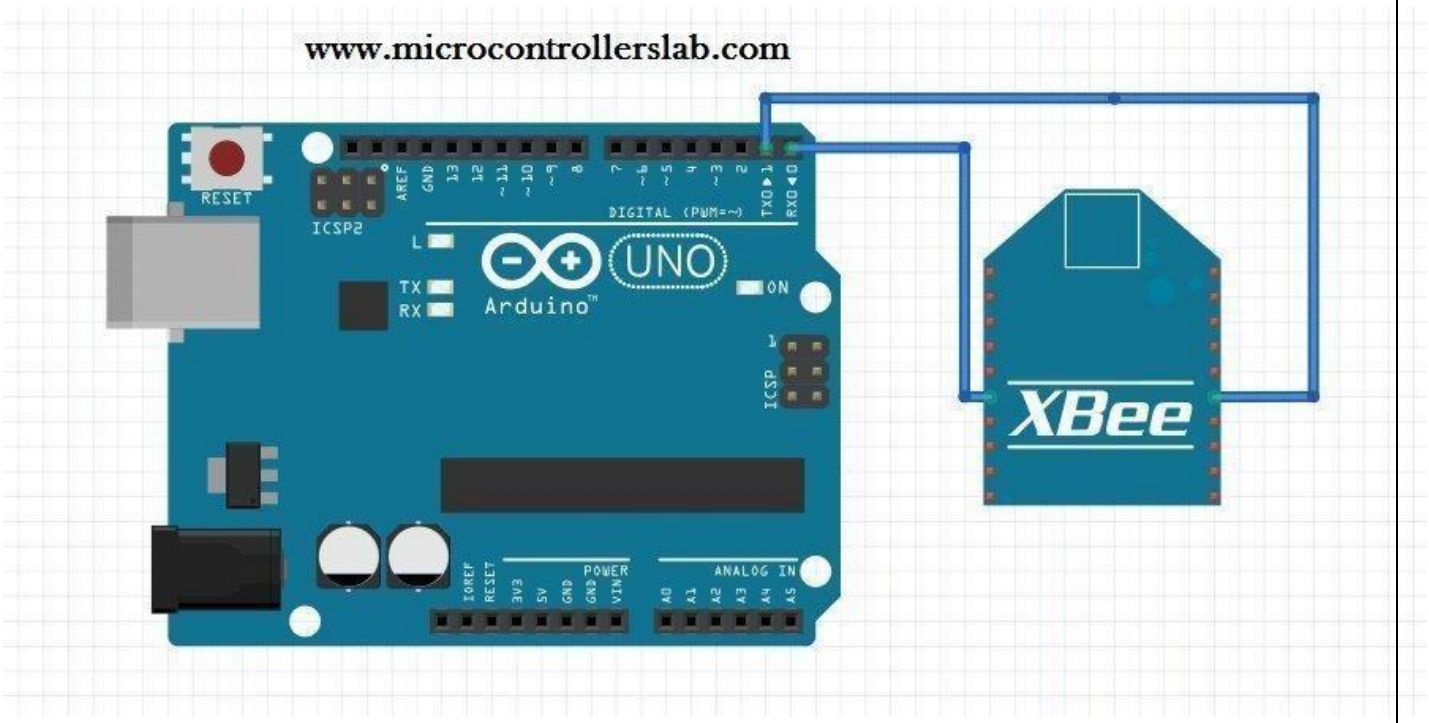
PROGRAM:
TRANSMITTER SIDE:

```
#include<SoftwareSerial.h>
SoftwareSerial mySerial(2,3); //rx,tx
void setup() {
        mySerial.begin(9600);
        Serial.begin(9600);
}

void loop() {
        mySerial.write('A');
        Serial.println('A');
        delay(100);
        mySerial.write('B');
        Serial.println('B');
        delay(100);
}
```

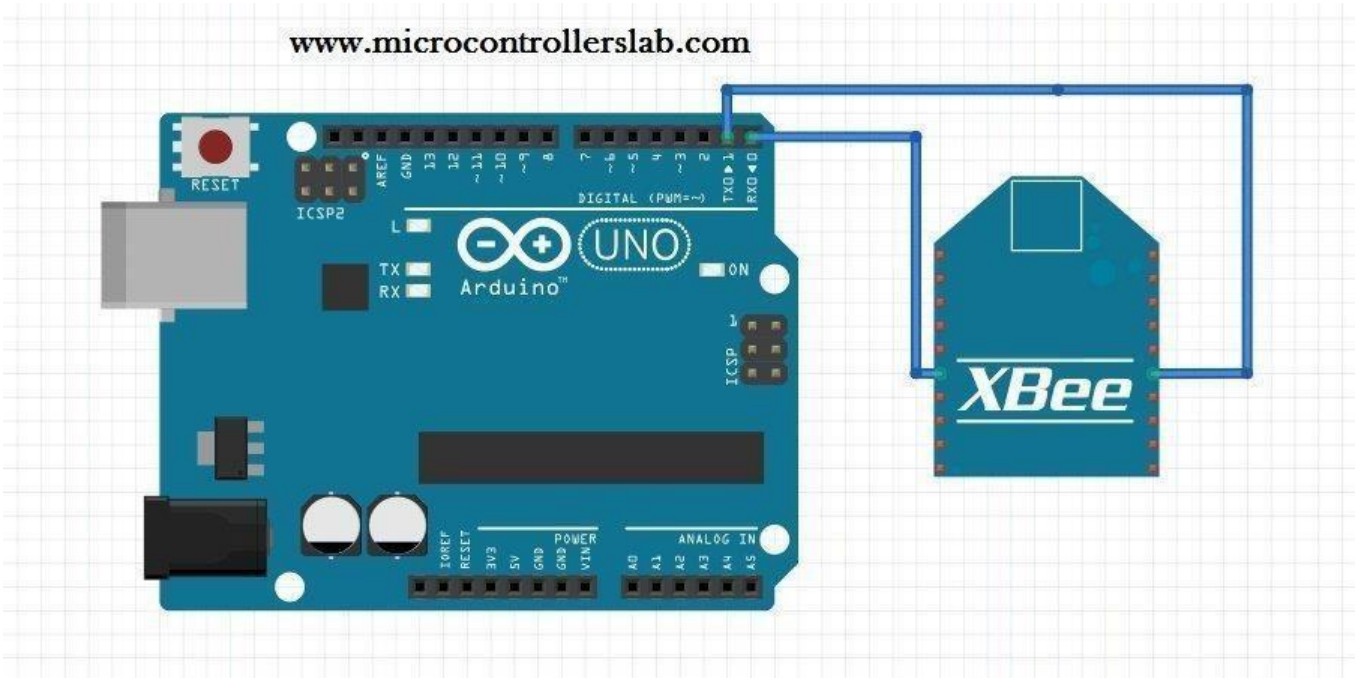
RECEIVER SIDE:

```
#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=='A')
         digitalWrite(4,HIGH);
         else if(data=='B')
         digitalWrite(4,LOW);
}
}
```

TRANSMITTER SECTION



RECEIVER SECTION



Applications of Zigbee

Some of the famous applications are:

- Wireless communication
- wirelessly controlled robot
- remote monitoring system
- Wireless home automation system
- wireless temperature sensor and many others.

RESULT:

EXP NO:13	INTRODUCTION TO THE RASPBERRY PI PLATFORM
DATE	

Introduction to Raspberry Pi Pico W:

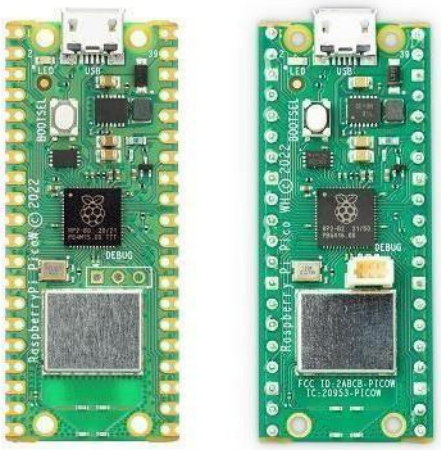
The Raspberry Pi Pico W is a compact and affordable microcontroller board developed by the Raspberry Pi Foundation. Building upon the success of the Raspberry Pi Pico, the Pico W variant brings wireless connectivity to the table, making it an even more versatile platform for embedded projects. In this 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

Raspberry Pi Pico W:

The Raspberry Pi Pico W is based on the RP2040 microcontroller, which was designed by Raspberry Pi in-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 IoTprojects, 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 programminglanguage 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

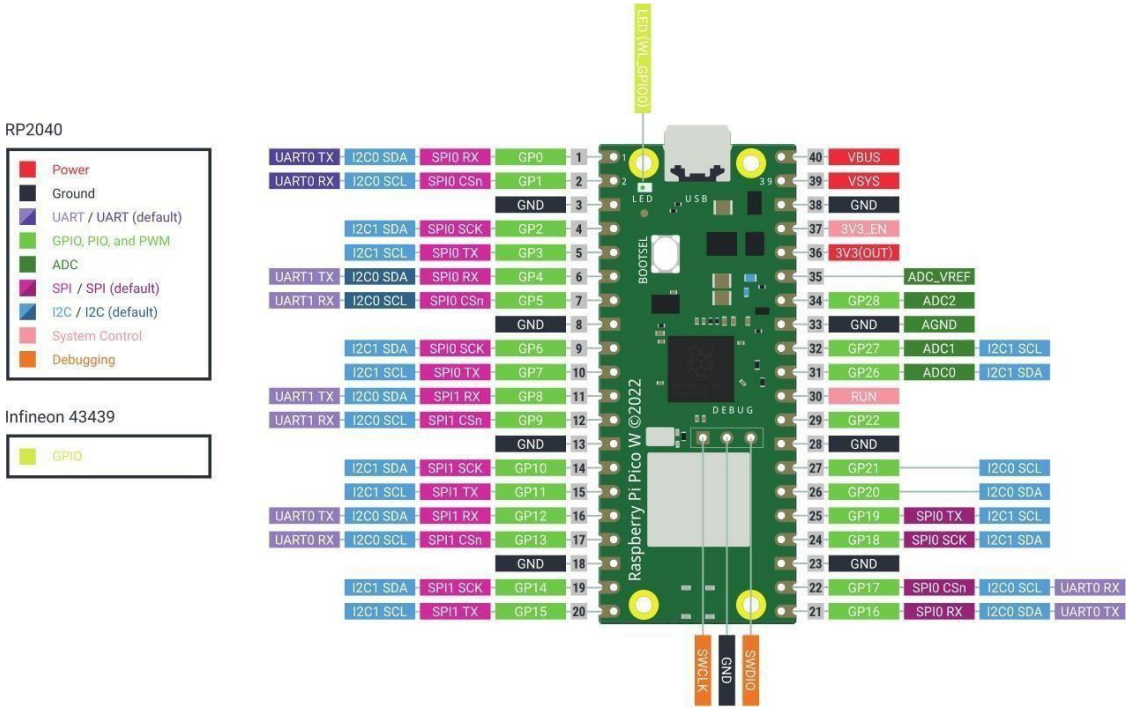
Pico W is compatible with C/C++ programming, allowing experienced developers to leverage the rich ecosystem of libraries and frameworks available.

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, shareknowledge, and collaborate on innovative projects.



The Raspberry Pi Pico W brings wireless connectivity to the popular Raspberry Pi Pico microcontroller board. 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. Whether you are a beginner or an experienced developer, the Raspberry Pi Pico W provides a user-friendly and flexible platform to bring your ideas to life and explore the exciting world of wireless IoT applications.

RESULT:

EXP NO:14	INTRODUCTION TO PYTHON PROGRAMMING
DATE	

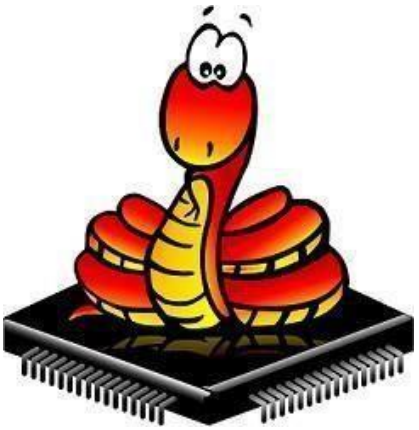
Getting Started with Thonny MicroPython (Python) IDE:

If you want to program your ESP32 and ESP8266 with MicroPython firmware, it's very handy to use an 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 and embedded 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 between Python 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 includes a 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, and Mac OS.

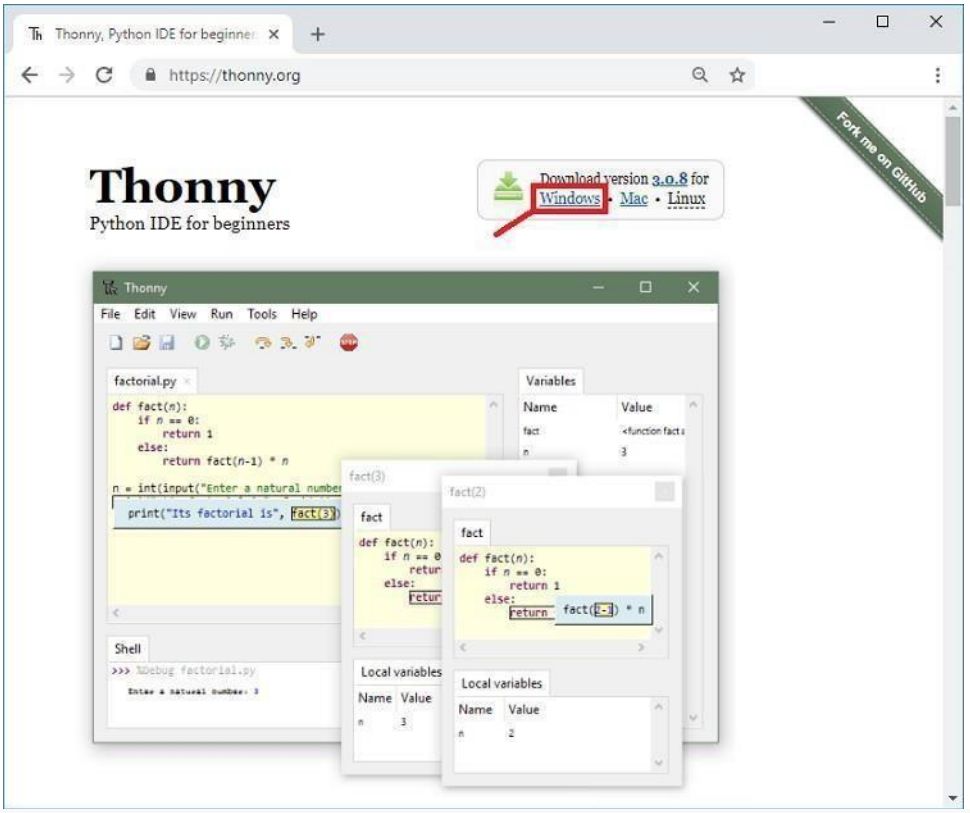
A) Installing Thonny IDE – Windows PC

Thonny IDE comes installed by default on Raspbian OS that is used with the Raspberry Pi board.

To install Thonny on your Windows PC, follow the next instructions:

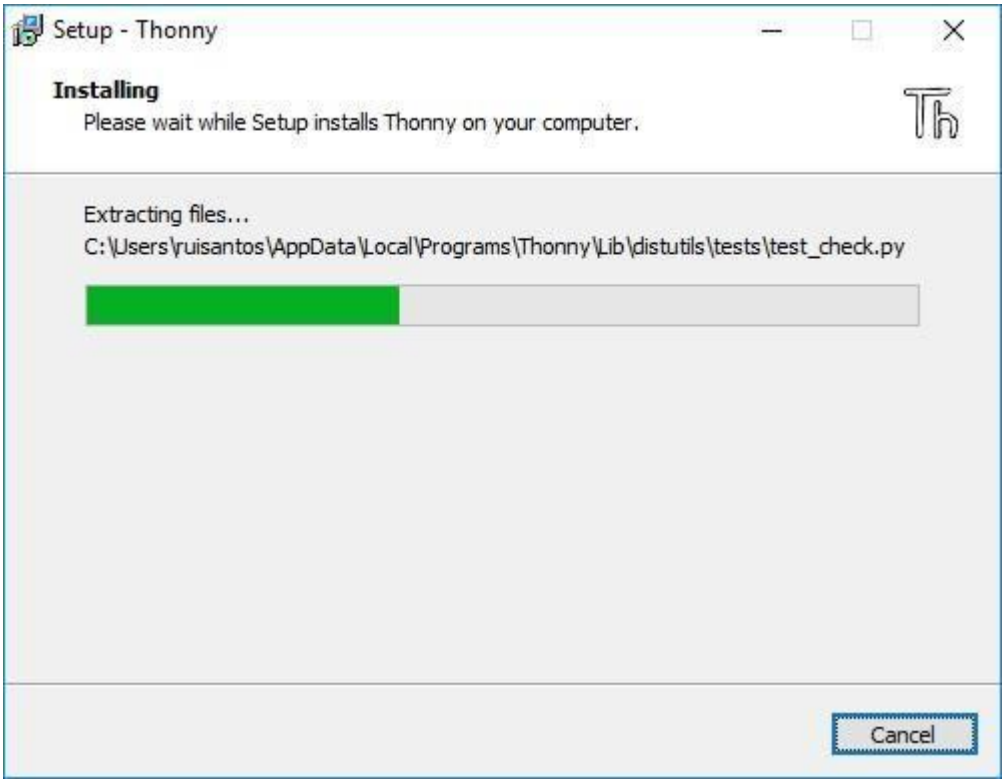
1. Go to https://thonny.org

2. Download the version for Windows and wait a few seconds while it downloads.

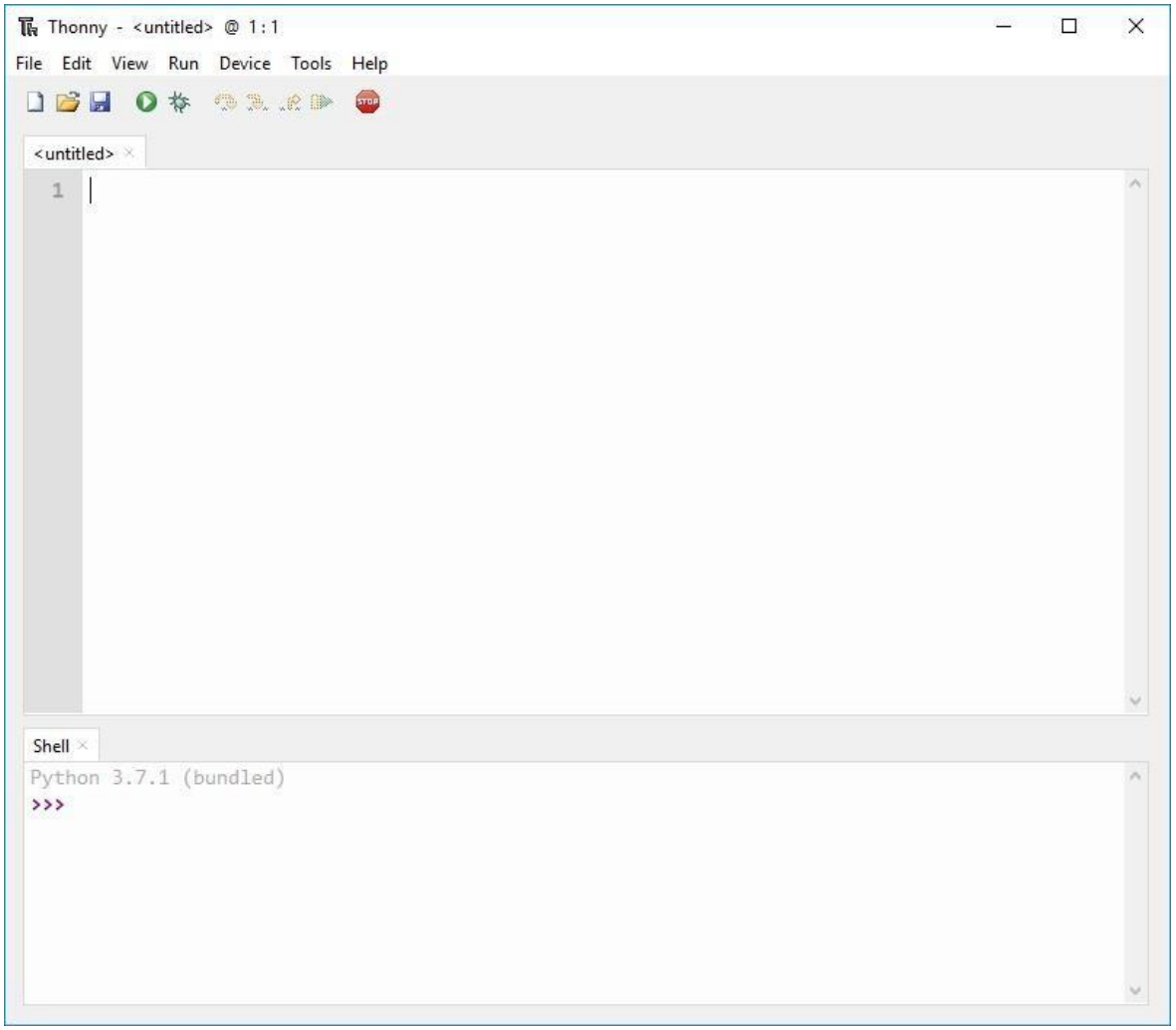


3. Run the *.exe* file.

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



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


Thonny - <untitled> @ 1 : 1
File Edit View Run Device Tools Help
<untitled>
1
Shell
Python 3.7.1 (bundled)
>>>

CONNECTIONS:

Pin	Development Board
GP16	**LED**

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)
```

CONNECTIONS:

Raspberry Pi Pico Pin	Raspberry Pi Pico Development Board (RGB)
GP16	**R**
GP17	**G**
GP18	**B**
GND	**COM**

RGB:

```
from machine import Pin
from time import sleep_ms,sleep
r=Pin(16,Pin.OUT)
y=Pin(17,Pin.OUT)
g=Pin(18,Pin.OUT)

while True:
    r.value(1)
    sleep_ms(1000)
    r.value(0)
    sleep_ms(1000)
    y.value(1)
    sleep(1)
    y.value(0)
    sleep(1)
    g.value(1)
    sleep(1)
    g.value(0)
    sleep(1)
```

CONNECTIONS:

Raspberry Pi Pico Pin	Raspberry Pi Pico Development Board
GP16	LED
GP15	SW1

SWITCH CONTROLLED LED:

```
from machine import Pin
from time import sleep
led=Pin(16,Pin.OUT)
sw=Pin(15,Pin.IN)
while True:
bt=sw.value()if
bt== True:
print("LED ON")
led.value(1)   sleep(2)
led.value (0) sleep(2)
led.value (1) sleep(2)
led.value(0)   sleep(2)
else:
   print("LED OFF")
sleep(0.5)
```

RESULT:

EXP NO:15	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 & 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	Ultrasonic sensor	1

PROCEDURE

To interface sensors with Raspberry Pi, you need to:

1. Identify which GPIO pins to use. Most sensors will connect to the Raspberry Pi GPIO directly or through an analog-to-digital converter (ADC).
2. Wire the circuit. The sensor needs to be wired to the GPIO to form a circuit.
3. Write code to read from the sensor.
4. Enable UART communication on the Raspberry Pi if you are using an ultrasonic sensor to read serial data.
5. Supply external power to the sensor when using an XL-Max Sonar or WR ultrasonic sensor with aRaspberry Pi

CONNECTIONS:

Pin	Development Board	IR Sensor Module
GP16	**BUZZER**	**-**
GP15	**-**	**OUT**
-	**5V**	**VCC**
-	**GND**	**GND**

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)
```

CONNECTIONS:

Pin	Development Board	Ultrasonic Sensor Module
GP16	**BUZZER**	**-**
GP15	**-**	**ECHO**
GP14	**-**	**TRIG**
-	**5V**	**VCC**
-	**GND**	**GND**

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)
```

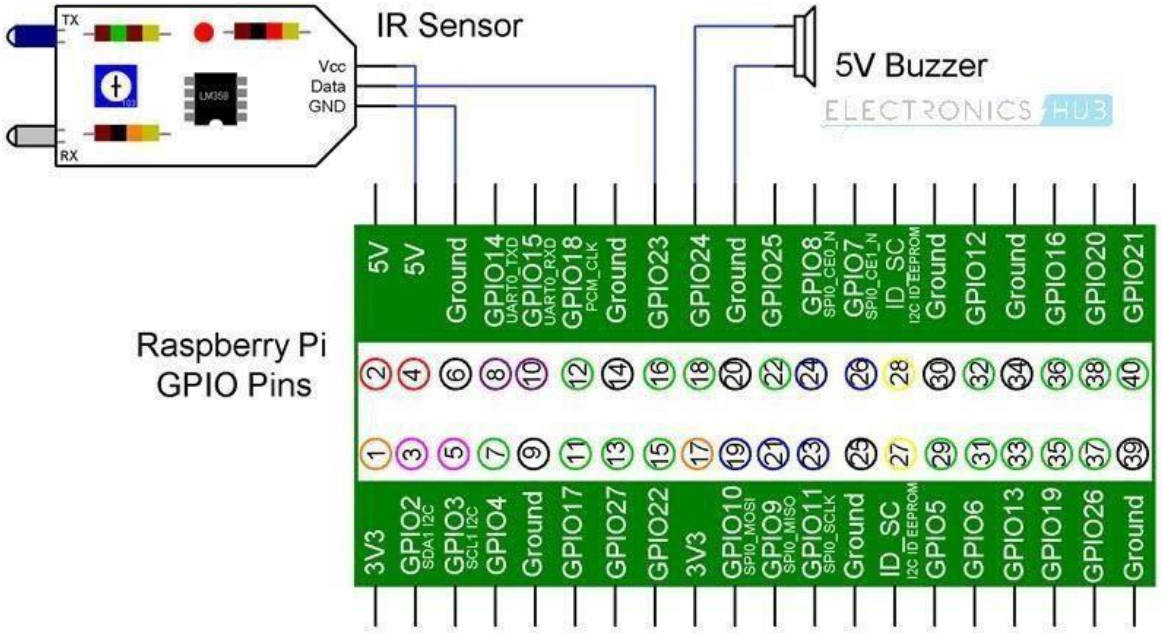


Figure. the connection diagram of Interfacing IR Sensor with Raspberry Pi

RESULT:

EXP NO:16	COMMUNICATE BETWEEN ARDUINO AND RASPBERRY PI
DATE	

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

To communicate between Arduino and Raspberry Pi, you can follow these steps:

1. Connect the Arduino to the Raspberry Pi through a USB cable.
2. On the Raspberry Pi side, use a simple USB connector.
3. On the Arduino side, use the USB port that you use to upload code from your computer to your board.
4. Use simple serial communication over USB cable to communicate between the two boards.
5. Upload the serial_test.ino code to your Arduino and run the serial_test.py Python code in Raspberry Pi.

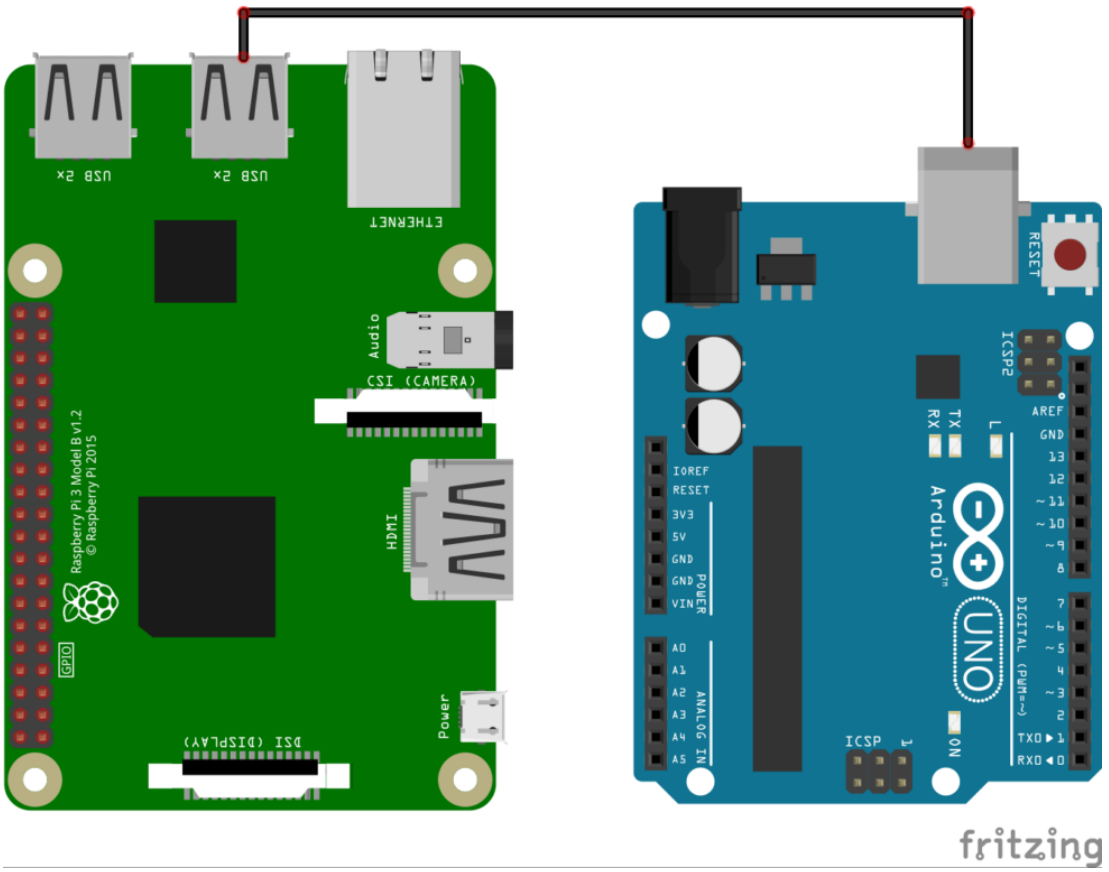


Figure: C**ommunicate between Arduino and Raspberry Pi**

CONNECTIONS:

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

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:

Pin	Development Board	
GP16	**LED**	**-**
VCC	**-**	**+5V**
GND	**-**	**GND**
GP1	**-**	**Tx**
GP0	**-**	**Rx**

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')
```

RESULT:

EXP NO:17	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 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 a platform to create and control devices where users can connect physical devices to the Internet and control them using a mobile app.

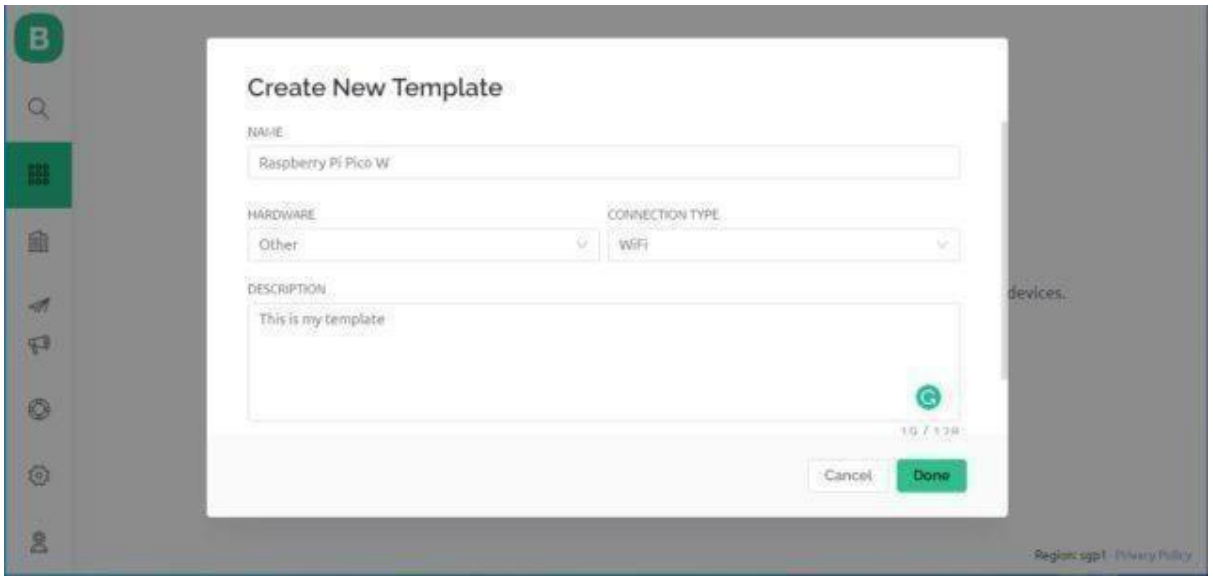
Setting up Blynk 2.0 Application

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:

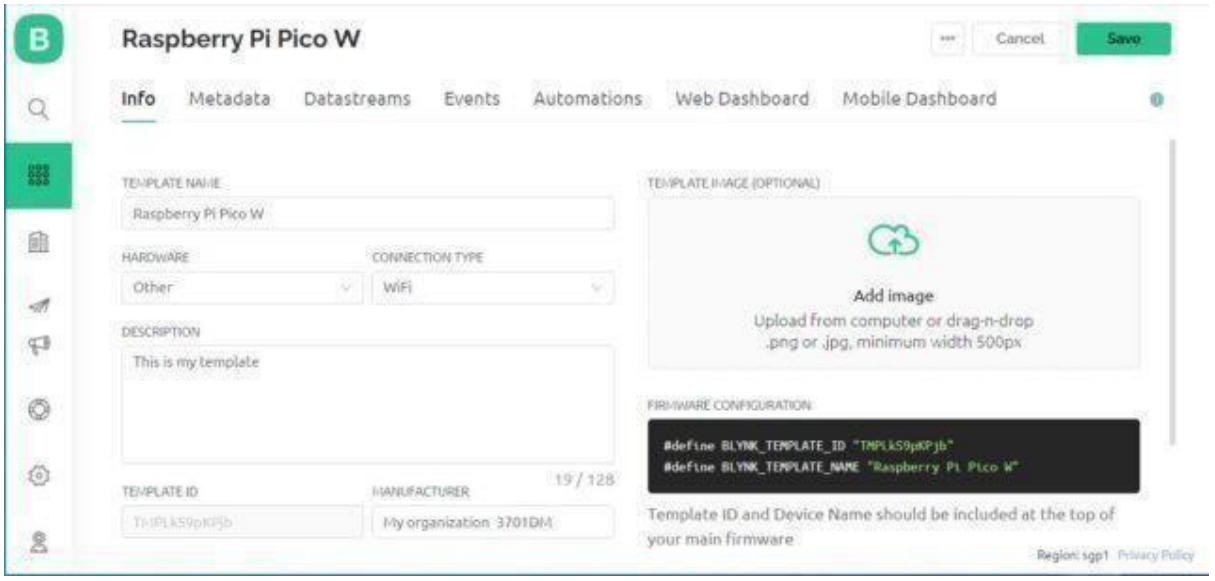
Step 1: Visit **blynk.cloud** and create a Blynk account on the Blynk website. Or you can simply sign in using the registered Email ID.

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

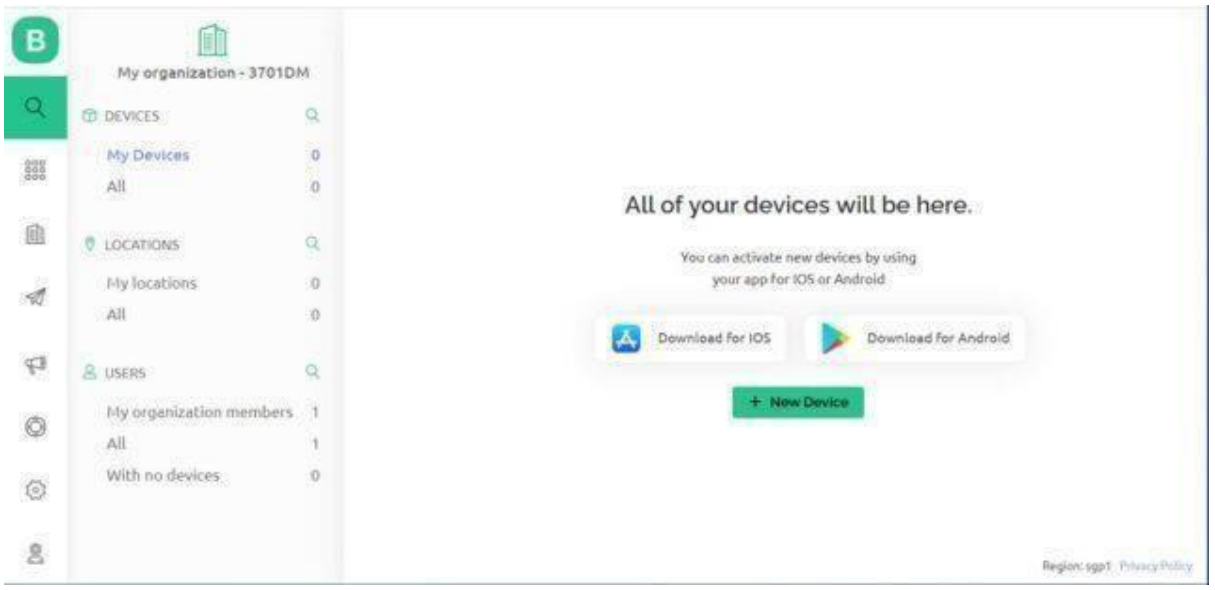
Step 3: Give any name to the Template such as Raspberry Pi Pico W. Select ‘Hardware Type’ as Other and ‘Connection Type’ as WiFi.



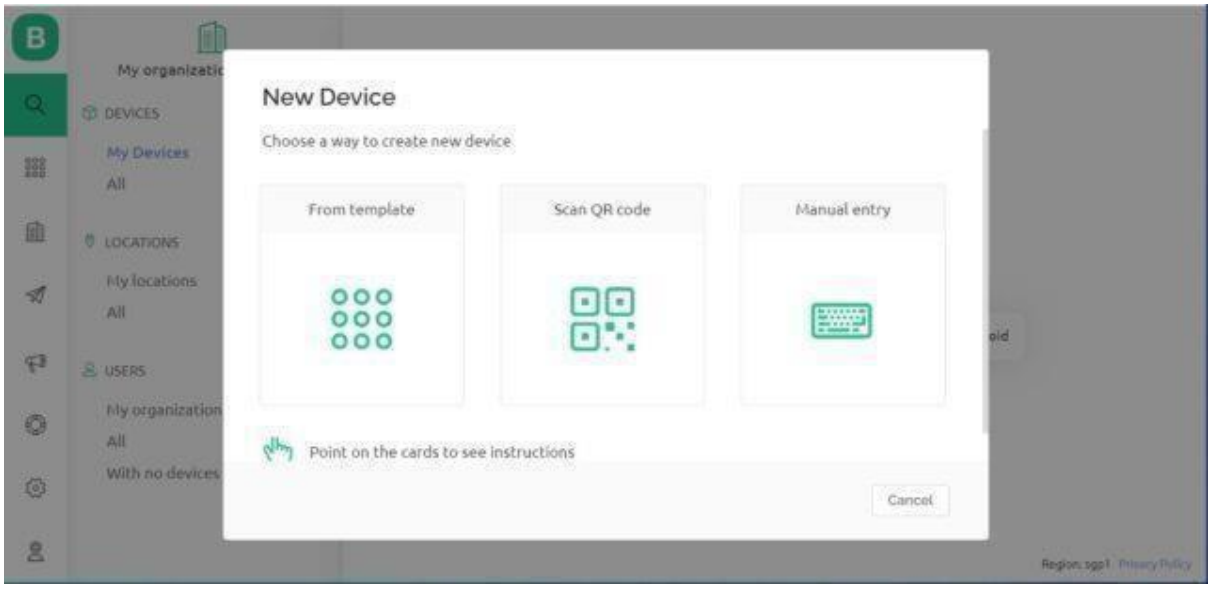
So a template will be created now.



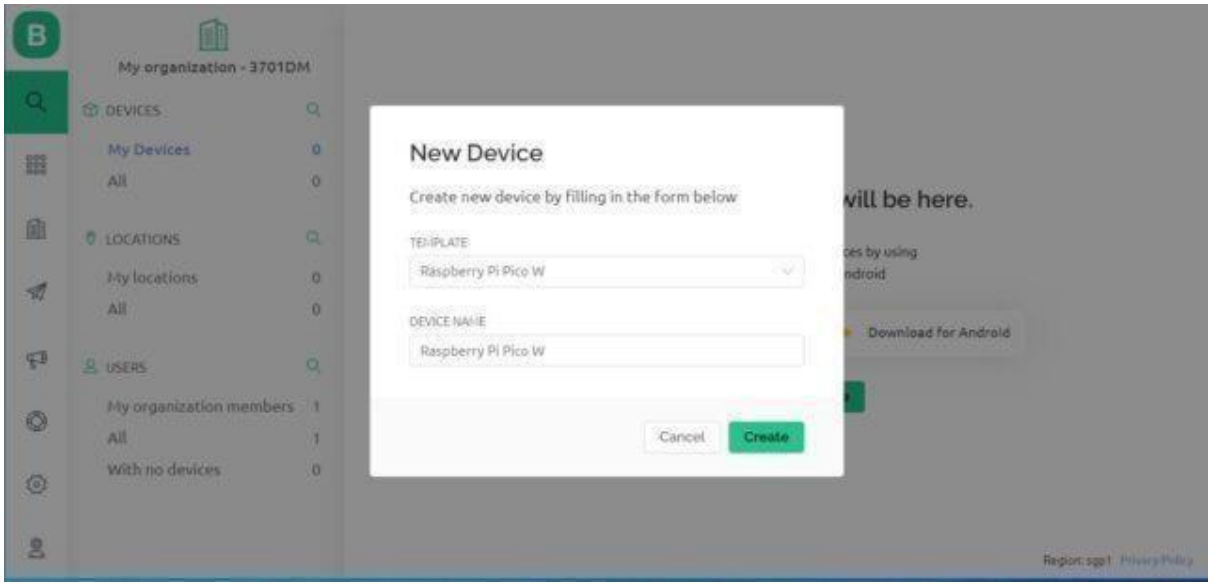
Step 4: Now we need to add a ‘New Device’ now.



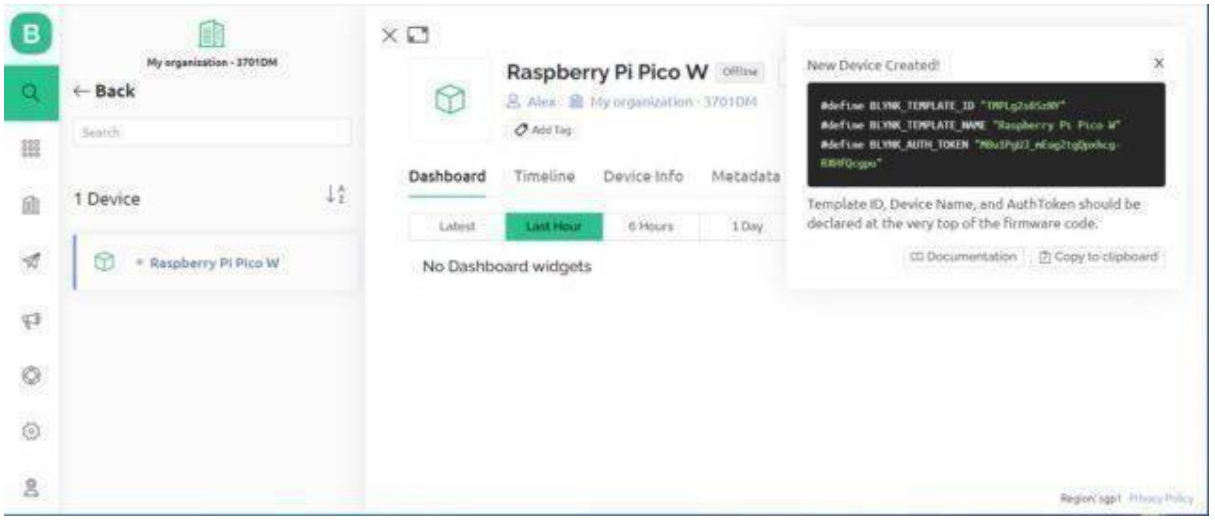
Select a New Device from ‘Template’.



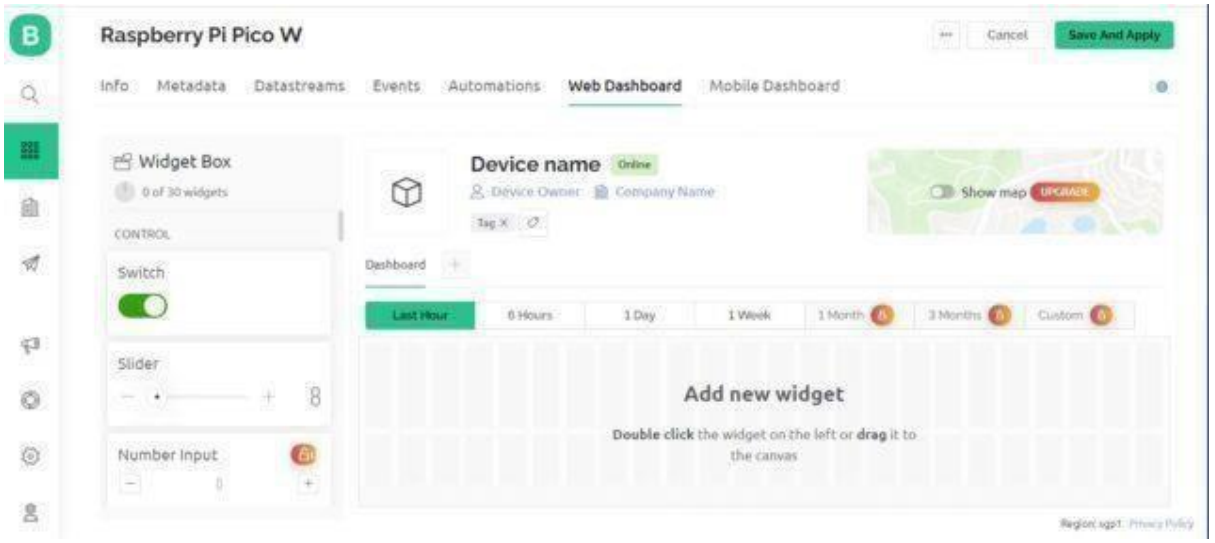
Select the device from a template that you created earlier and also give any name to the device. Click on Create.



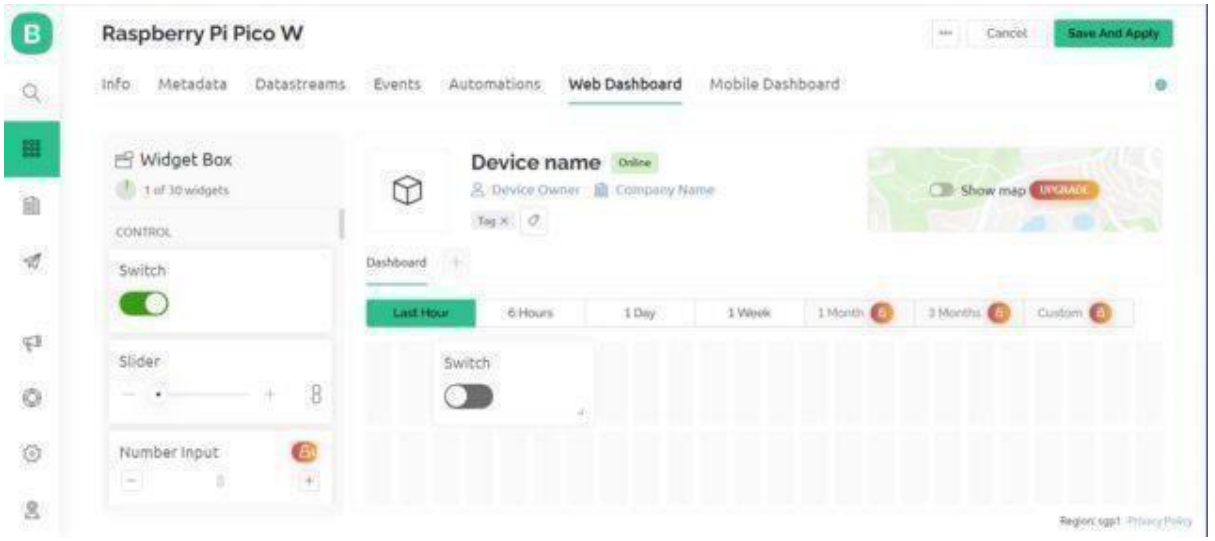
A new device will be created. You will find the Blynk Authentication Token Here. Copy it as it is necessary for the code.



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



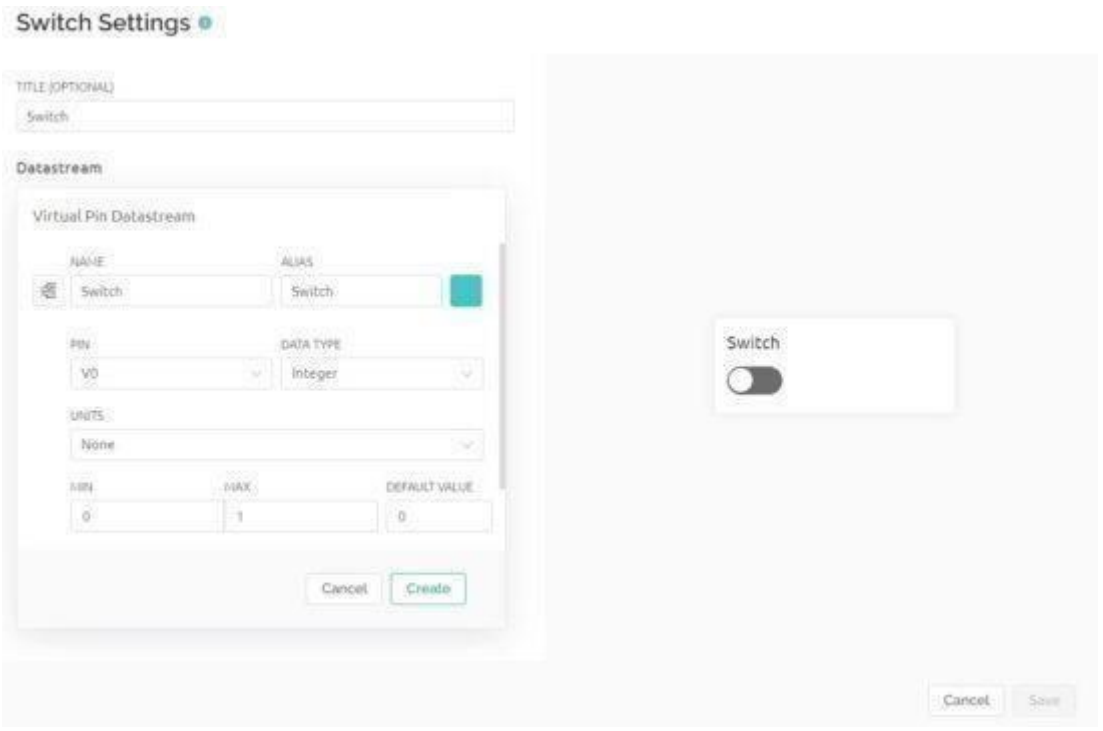
From the widget box drag a switch and place it on the dashboard screen.



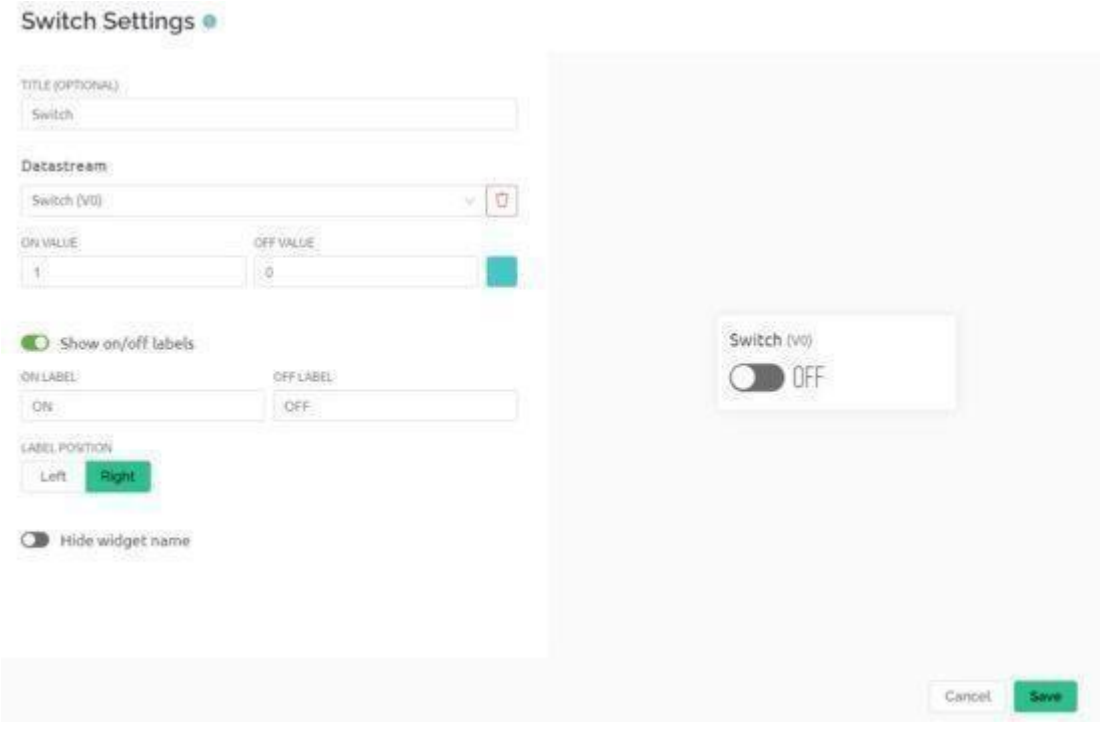
Step 6:

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.



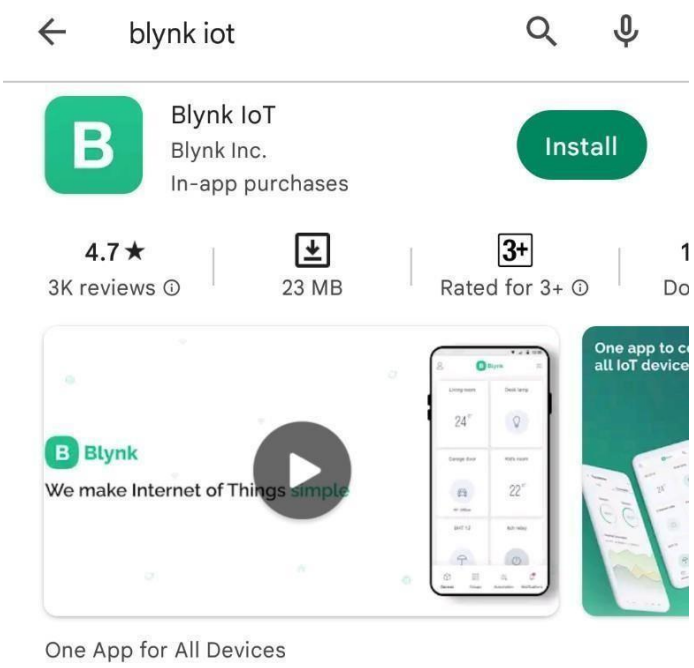
Configure the final steps again.



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:

EXP NO:18	Log Data using Raspberry PI and upload it to the cloud 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	Jumper Wires	1
4	Micro USB Cable	1

PROCEDURE

CONNECTIONS:

Raspberry Pi Pico Pin	Raspberry Pi Pico Development Board	LCD Module
-	5V	VCC
-	GND	GND
GP0	-	SDA
GP1	-	SCL

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'

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

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

"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)
```

RESULT

83

Ex No:	DESIGN AN IOT BASED SYSTEMS
Date:	

IOTbased Health Monitoring System using Arduino

AIM:

In this project, a system for 24×7 human health monitoring is designed and implemented.

DESCRIPTION:

In this system, the Arduino Uno board is used for collecting and processing all data. Different sensors are used for measuring different parameters. All this data is uploaded to thingspeak for remote analysis. An ESP8266 module is used for connecting to the internet. A solar power system is provided for powering all the sensors.

Abstract

This project introduces a wireless health monitoring system that can monitor a human 24×7. This system consists of a number of the part. Controlling and data processing is done through the Arduino Uno board, all the sensors are connected to Arduino UNO.

Through this system, we can measure ECG, heartbeat, BP, and spo2. Through sensors, it is possible to measure all these values. Here all the sensors are powered using a solar power system. All these analog sensors can be connected to Arduino through any of the six analog pins. These values are then used for detecting any critical situation. In the case of a critical situation, an alert can be given as a message.

Also, it is possible to monitor the person's health from any location in the world through the Thingspeak cloud. Data from sensors is uploaded to the Thingspeak periodically without any interruption if the internet is available. Here ESP8266 wifi module is used for connecting Arduino to the internet.

Introduction

Health is the most important part of any human's life without health it is useless to any treasure of life. Most humans live a busy life in which going to a doctor for weekly or even monthly checkup is an impossible task. Without monitoring your

health it is not possible to whether you are a healthy or sick person. This problem leads to the design of a product which monitors your health every day without going to a doctor. In this paper, a system is designed as a prototype for monitoring alerting based on the health of a person. This system is fully automated little or no human help is needed. Any doctor can monitor this person from anywhere through the internet.

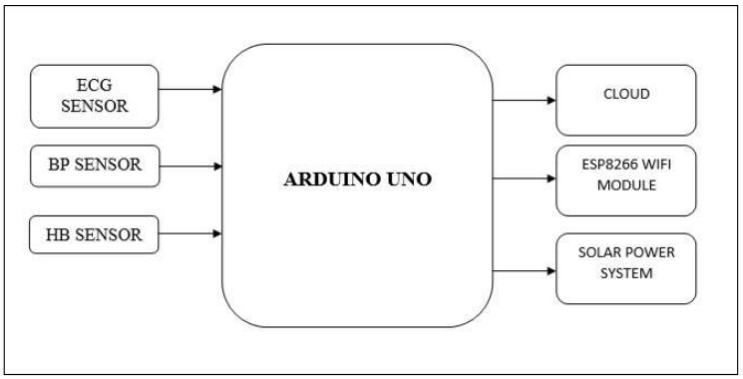
Existing System

- Diagnosing with the help of a doctor
- Conventional devices that can only measure a particular parameter
- Devices that have to be connected invasively to get measurements
- No automated system exists
- Smart watches are expensive and not specifically for healthcare

Proposed System

- In this project, a system for 24×7 human health monitoring is designed and implemented
- In this system, the Arduino Uno board is used for collecting and processing all data
- Different sensors are used for measuring different parameters
- All this data is uploaded to thingspeak for remote analysis
- An ESP8266 module is used for connecting to the internet
- A solar power system is provided for powering all the sensors

BLOCK DIAGRAM



BLOCK DIAGRAM DESCRIPTION

- Arduino Uno is the controller board which is a heart-whole system
- All the different analog sensors are connected to Arduino through analog pins
- Here the ESP8266 WiFi module connects the whole system to a WiFi network
- Data from sensors are uploaded to the cloud.

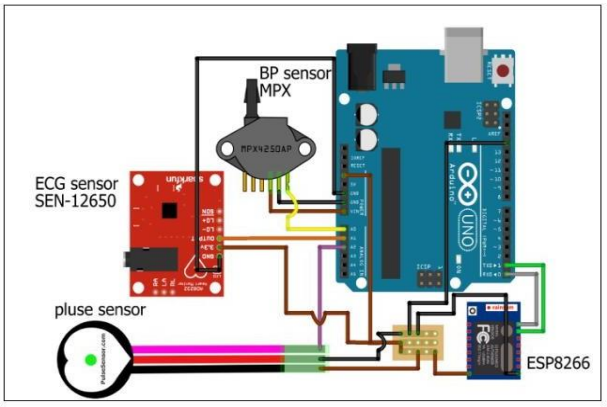
HARDWARE REQUIRED

- Arduino Uno
- ECG SENSOR
- Heartbeat Sensor
- BP sensor

SOFTWARE REQUIRED

- Arduino IDE

CIRCUIT DIAGRAM



CONCLUSION

This system is very effective in monitoring a person's health continuously because it is fully automated. It can be tested very easily with any person. This system is a very good example of remote health monitoring.

RESULT: