Build an IoT Water Meter!

Index

Aim

Concept

Components

Connections

Software

- For Dexter
- ESP32 & Ubidots Setup

Real-world Application

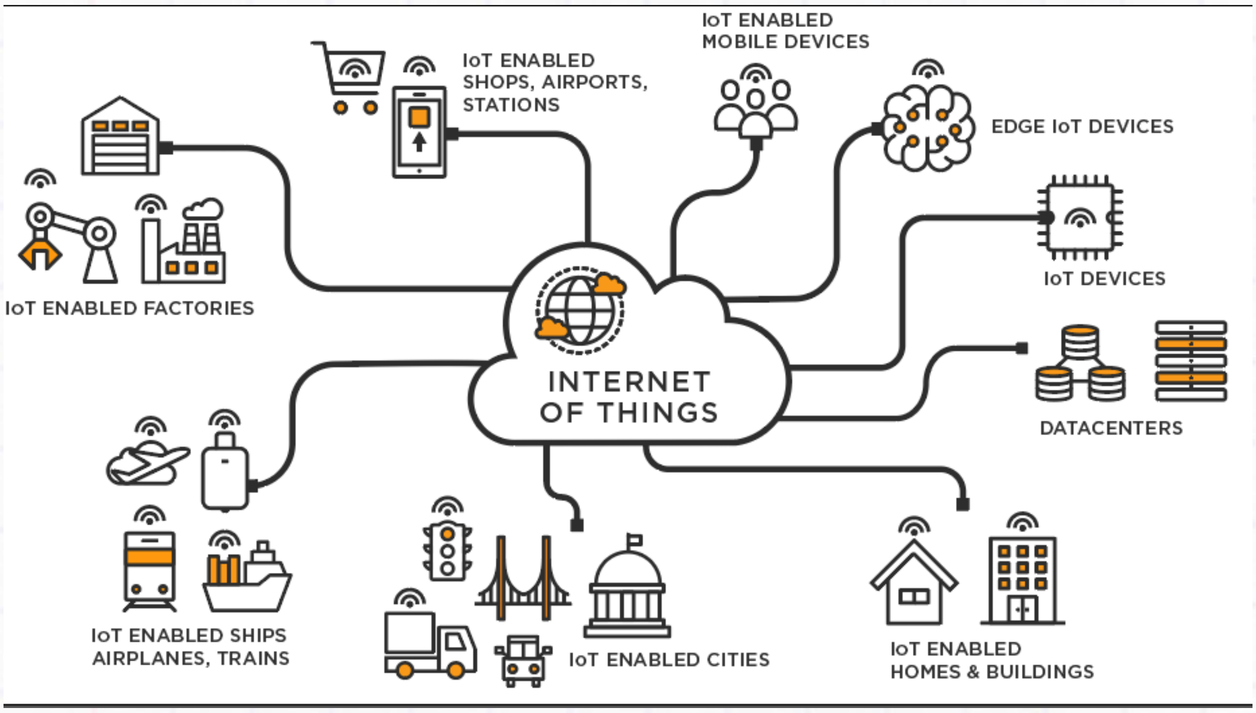
Aim

Build an IoT (Internet of Things) Water Flow Meter that measures water flow rate and consumption in real-time and track this data remotely on a web IoT platform.

Concept

What is IoT?

IoT is a system of interconnected computing devices that use internet protocols to communicate and transfer data. This allows remote devices to communicate amongst each other without the need for human involvement,

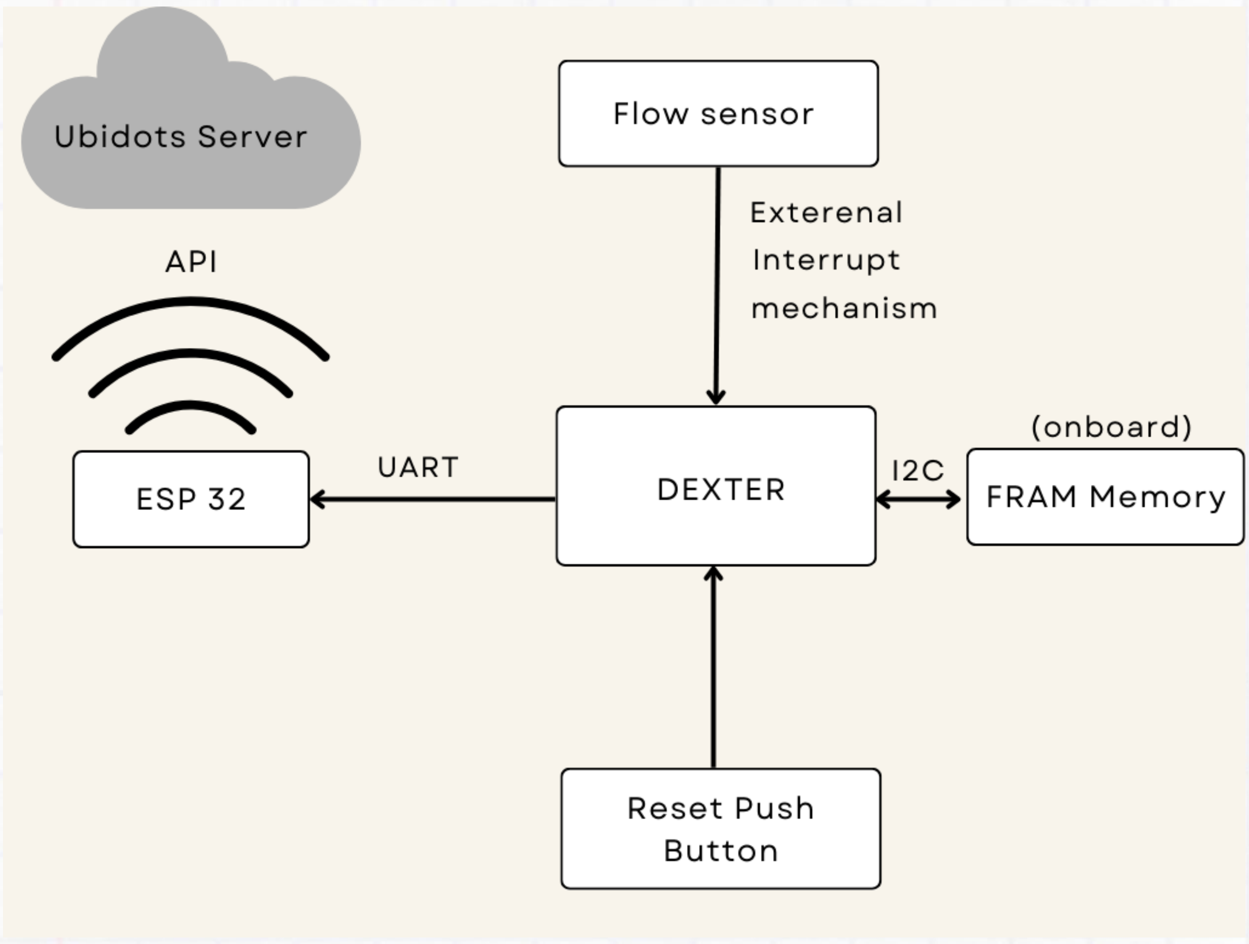

IoT ENABLED MOBILE DEVICES
IoT ENABLED SHOPS, AIRPORTS, STATIONS
EDGE IoT DEVICES
IoT ENABLED FACTORIES
IoT DEVICES
INTERNET OF THINGS
DATACENTERS
IoT ENABLED SHIPS AIRPLANES, TRAINS
IoT ENABLED CITIES
IoT ENABLED HOMES & BUILDINGS

How this project works

In this project, a flow sensor is used to measure the flow rate of any fluid passing through. This value is sent to the Dexter board using an External interrupt mechanism - which constantly measures the flow of the fluid.

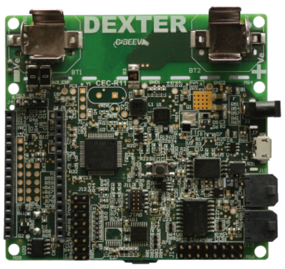
Through the code written to the Dexter, we calculate the values of Flow rate and Water accumulated. Then, using I2C protocol, we write the values onto the Dexter's on-board FRAM memory. A Push Button is also provided to reset the water accumulated reading in the FRAM memory.

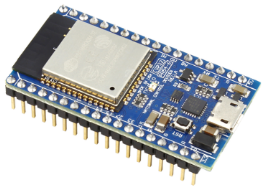
Next, we transmit the data stored in the FRAM to the ESP32 Wifi module via the UART interface. The ESP32 Wifi module then sends the data to a Ubidots cloud server using an API. The water flow data can then be viewed in real-time on the Ubidots web dashboard.



Components (*Provided in kits sent by Build Lab - IITM)

1. Dexter board
2. ESP32 Wifi module
3. Breadboard
4. YF-S201 Flow sensor
5. Push Button
6. PVC tubing
7. Jumper wires
8. 2 USB cables

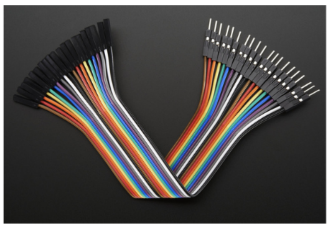

Dexter board

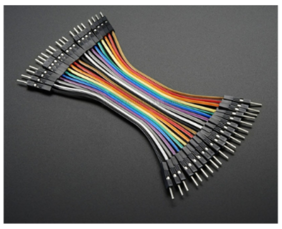

ESP32 Wifi module

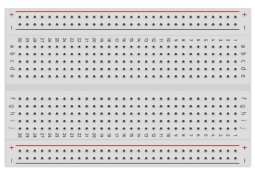
YF-S201 Flow sensor

USB cable

Push button


Male to female Jumper wires


Male to male Jumper wires


Breadboard

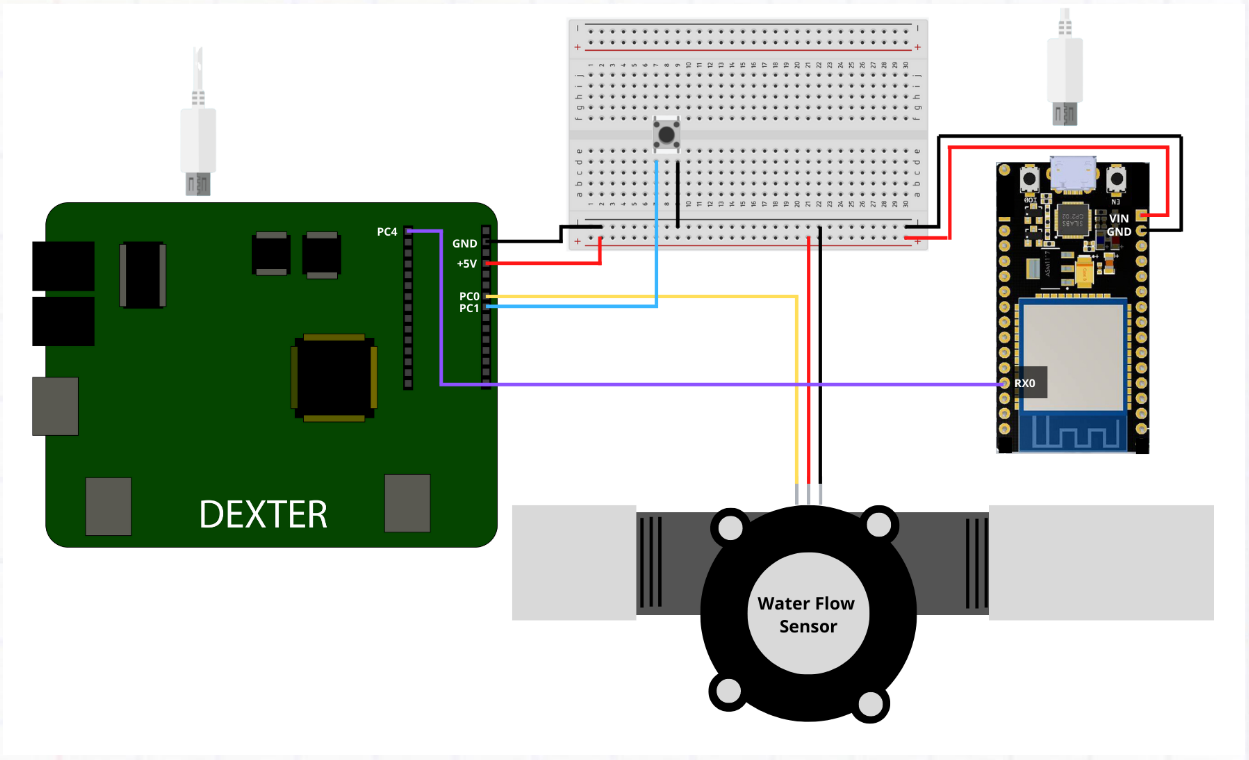
PVC tubing

Connections

Safety tip: Always ensure that the connections to the components are correct and completed before connecting the power supply to the Dexter board.

Circuit Diagram

- **NOTE**: Before starting the connections, verify using a multimeter that all the jumper wires are working. Also ensure that the connections are strong, else the setup may not work.

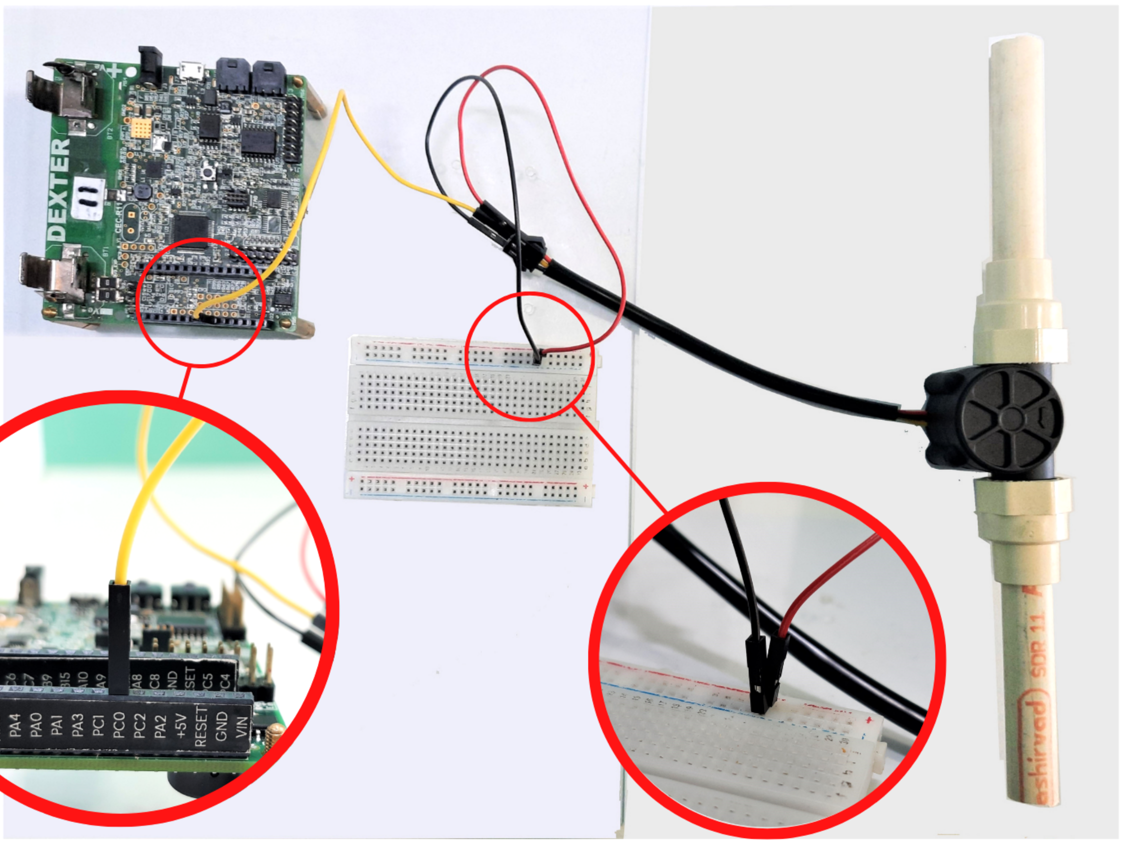


Detailed Connection Steps

Step 1

Take 3 male-to-male jumper wires and connect the pins of the YF-S201 Flow sensor (**red** - voltage, **black** - ground, **yellow** - output) as shown.

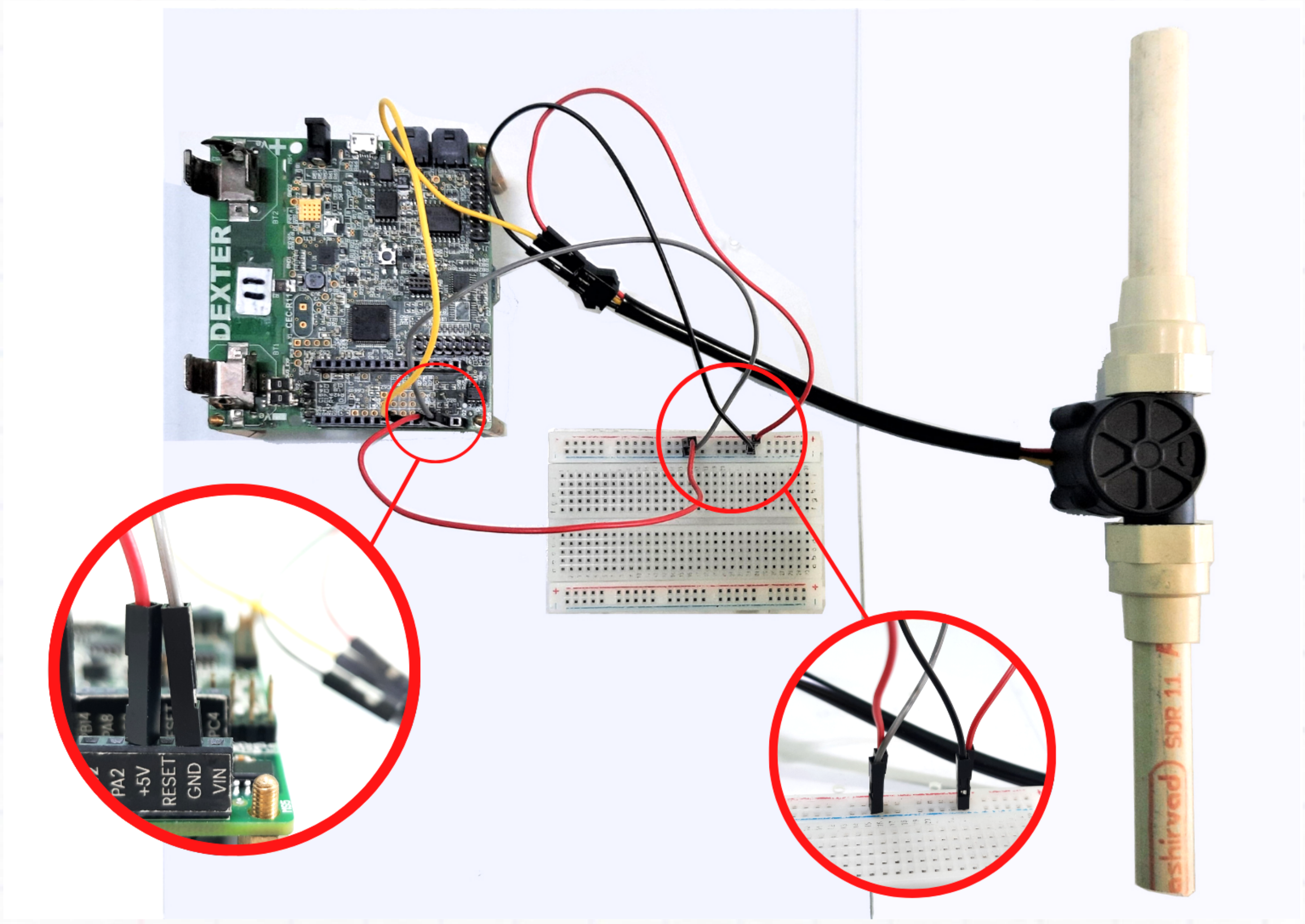
- **Red cable** (voltage) to **Red line** of breadboard
- **Black cable** (ground) to **Blue line** of breadboard
- **Yellow cable** (output) to **PC0** of the Dexter



Step 2

Take 2 more male-to-male jumper wires and connect them as follows:

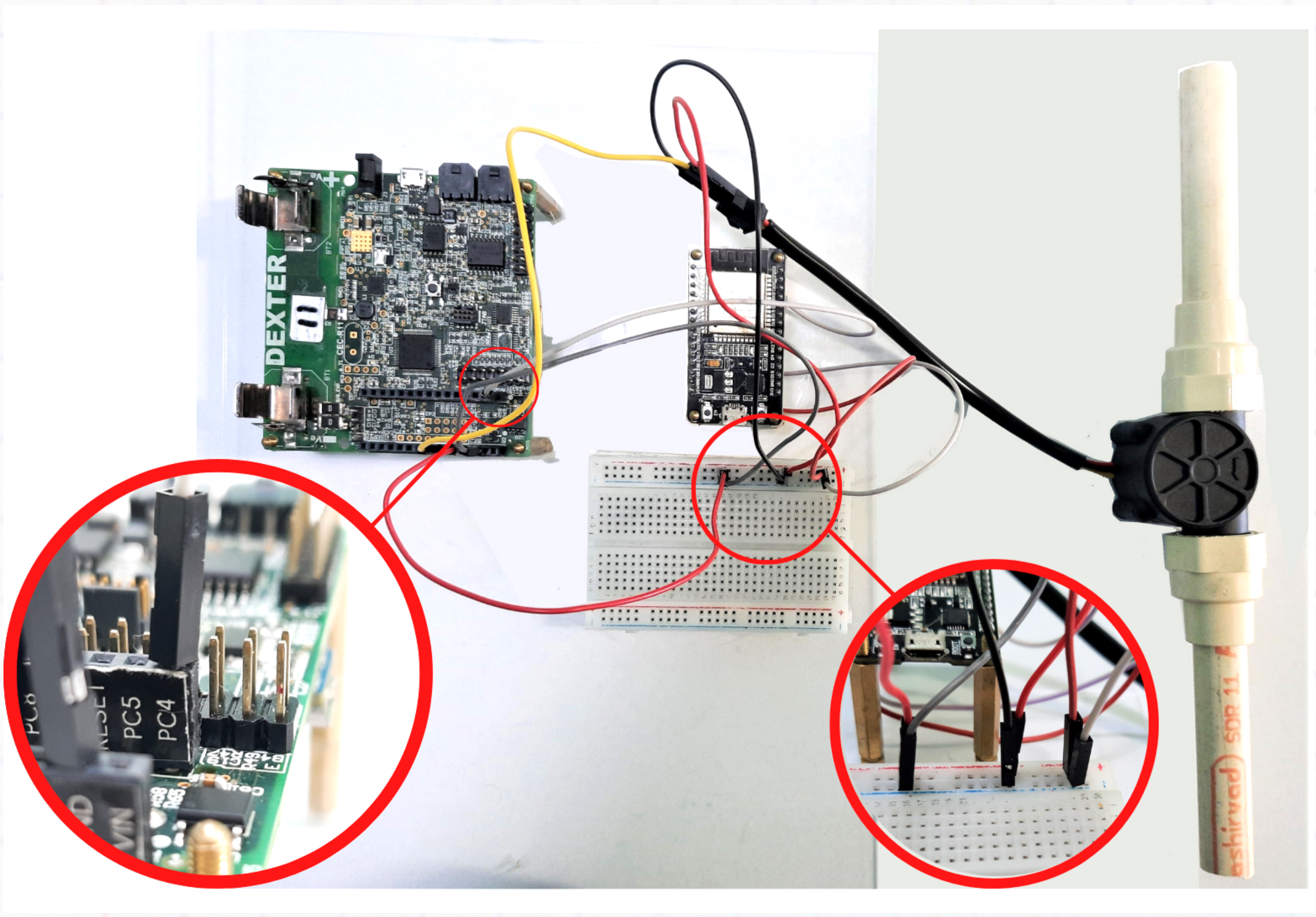
- **+5V** of Dexter to **Red line** of the breadboard
- **GND** of Dexter to **Blue line** of the breadboard



Step 3

Take 3 male-to-female jumper wires and connect the pins (**VIN**, **GND**, **RX0**) of the ESP32 Wifi module:

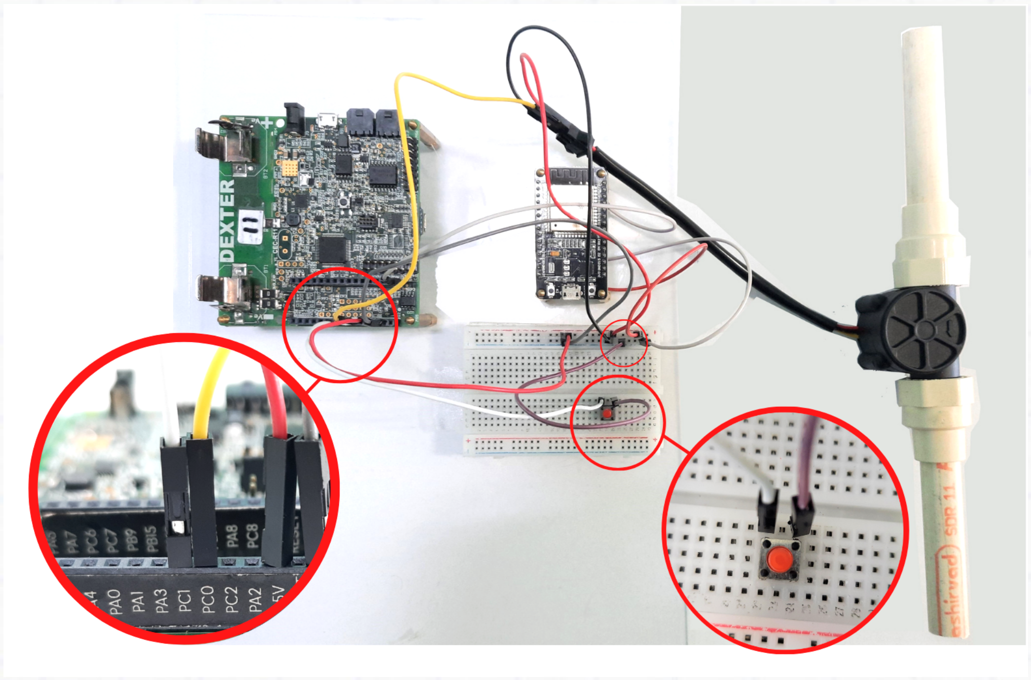
- **VIN** to **Red line** of the breadboard
- **GND** to **Blue line** of the breadboard
- **RX0** to **PC4** of the dexter board



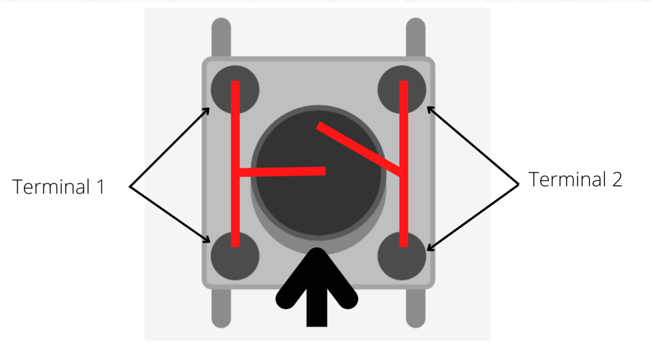
Step 4

Take 2 male-to-male jumper wires and connect the terminals of the Push button:

- One terminal to **blue line** of breadboard
- One terminal to **PC1** of the Dexter

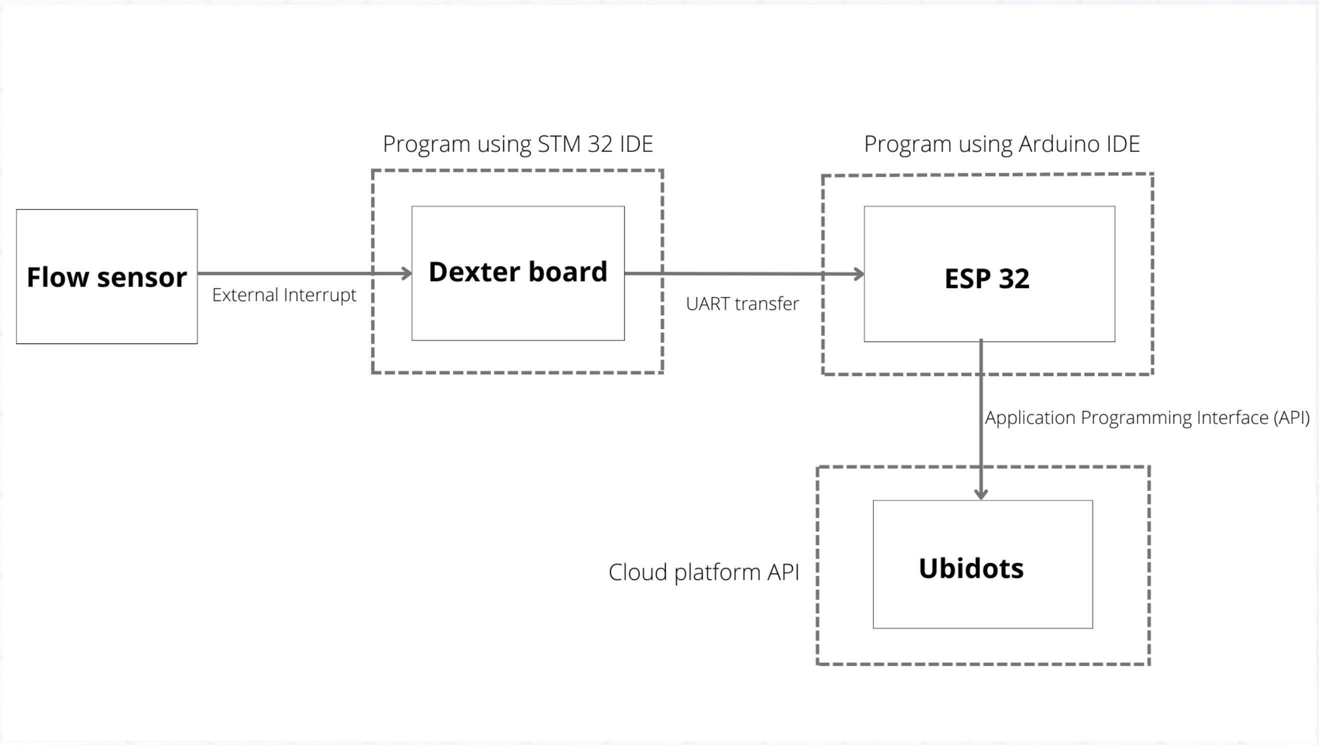


Note:



Pushbutton-
Connects two sides
when pressed

Software



For Dexter

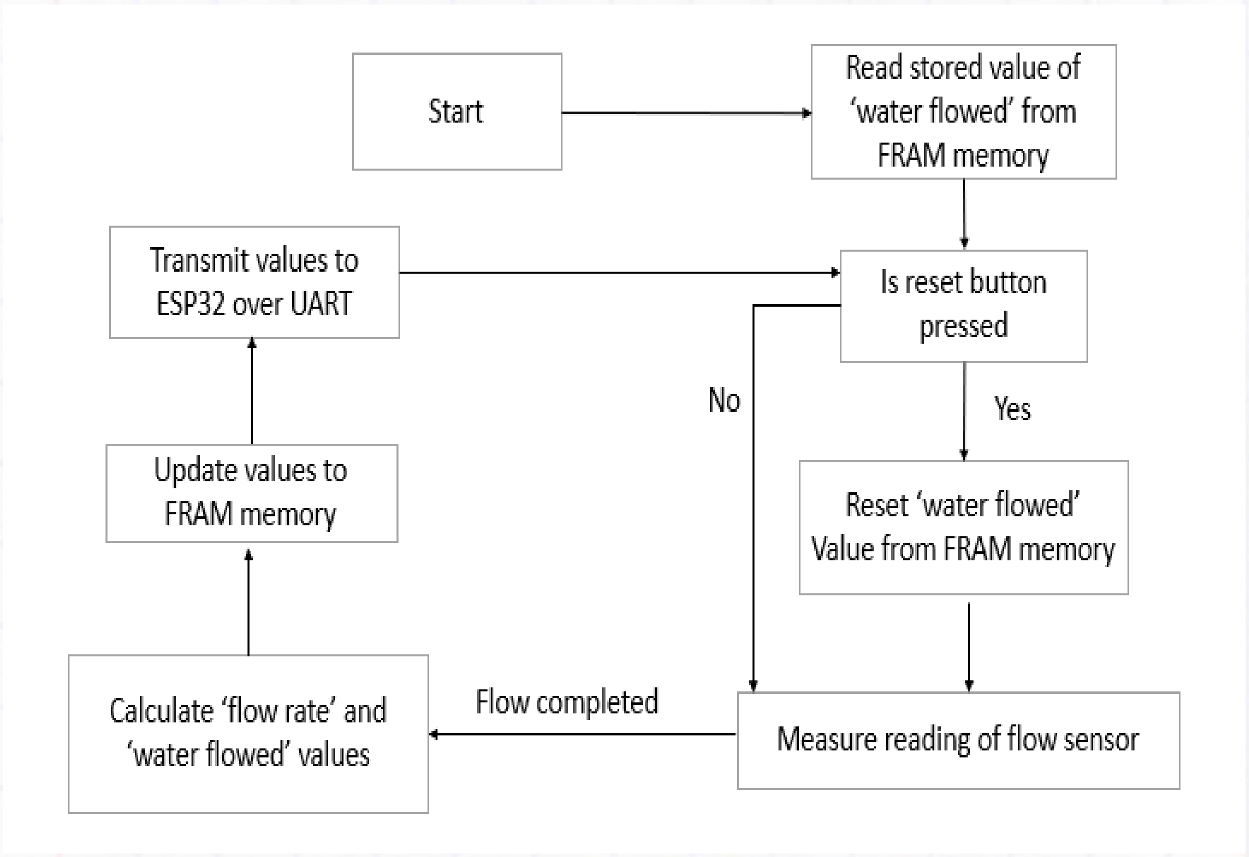
Downloads & Installation

1) Download the Project Workspace file **Dexter_Flow_Meter.zip**' given in the project page on the Build Club website.
2) In the **Workspace** folder in your C: drive, create a new folder named '**IoT_Water_Meter**'.
3) Now, as done in every project: i) Launch the STM IDE, ii) Select the **IoT_Water_Meter** folder as workspace, iii) Import the ZIP file **Dexter_Flow_Meter.zip**, iv) Navigate to **app.c**.

NOTE: Before getting into the software for the project, it is important to understand the concept behind it and get an overview of how the project works. For this reason, we recommend revisiting the **Concept** section at the start of the manual, where we explain how this project works.

Flowchart of the Code

This flowchart diagram captures the flow of logical steps needed to implement the IoT Water Flow Meter project. Once this is firmly understood, code can be easily written to implement the project and make it work.


Start
Read stored value of 'water flowed' from FRAM memory
Transmit values to ESP32 over UART
Is reset button pressed
No
Yes
Update values to FRAM memory
Reset 'water flowed' Value from FRAM memory
Calculate 'flow rate' and 'water flowed' values
Flow completed
Measure reading of flow sensor

Variables & Functions

Variables:

1) **total_water_flowed**

   Stores the value of total water that has flowed through the sensor.

2) **memory_reset_flag**

   Stores the status of the Reset push button and is used to check if total_water_flowed value stored in the memory needs to be reset.

3) **flow_rate**

   Stores the value of the instantaneous flow rate of water through the sensor.

4) **pulse_count**

   Counts the number of pulses generated in real-time during the flow period.

5) **cumulative_count**

   Counts the cumulative number of pulses generated across a flow period. It is equal to the pulse_count variable integrated over the period of a single flow.

6) **Flow_sensor & Reset_button**

   These two variables store the pin values of the flow sensor and push button. They are used in the Detect_interrupts function.

Functions:

1) **Read_from_FRAM();**

   Reads the total_water_flowed value stored in the FRAM memory.

2) **Reset_FRAM();**

   Resets the total_water_flowed value stored in the FRAM memory.

3) **Check_flow_status();**

   Waits for flow to be completed.

4) **Increment_water_reading();**

   Once flow is completed, the water flowed reading is incremented. This value is written to the FRAM memory.

5) **Calculate_flowrate();**

   Calculates the instantaneous flow rate of water through the sensor. This value is also written to the FRAM memory.

6) **UART_transmit(**flow value, water value**);**

   Transmits the values of **flow_rate** and **total_water_flowed** to the ESP32 device via UART communication interface.

Implementing the Code

Let us convert our flowchart of steps into working code. Use the corresponding functions and code shown below for each step.

1) Read stored value of total water flowed from FRAM memory:

```
total_water_flowed = Read_from_FRAM();
```

2) After this point, the code should enter into an infinite while loop:

```
while(1)
{
        // Rest of code
}
```

3) If Reset Button is pressed, reset value in the memory:

```
if(memory_reset_flag == 1)
{
        Reset_FRAM();
}
```

4) Measure flow sensor and wait for completion:

```
Check_flow_status();
```

5) Calculate flow rate & increment the water reading values and write them to the FRAM memory:

```
Increment_water_reading();
```

Calculate_flowrate();

6) Transmit values to ESP32 via UART:

UART_transmit(flow_rate, total_water_flowed);

After implementing these code commands in the order specified, we need to write a function to measure the flow rate from the sensor and read the reset push button. Since both use the External interrupt mechanism, we will handle them in a single function.

Below the **App()** function in **app.c**, you will find an empty function named **Detect_interrupts** with a single parameter called **input_device**. Write the code for this function by implementing the following algorithm of steps:

(Note: The **input_device** parameter can be equal only to either the **Flow_sensor** or **Reset_button** variables mentioned before.)

Pseudocode for Detect_interrupts function:

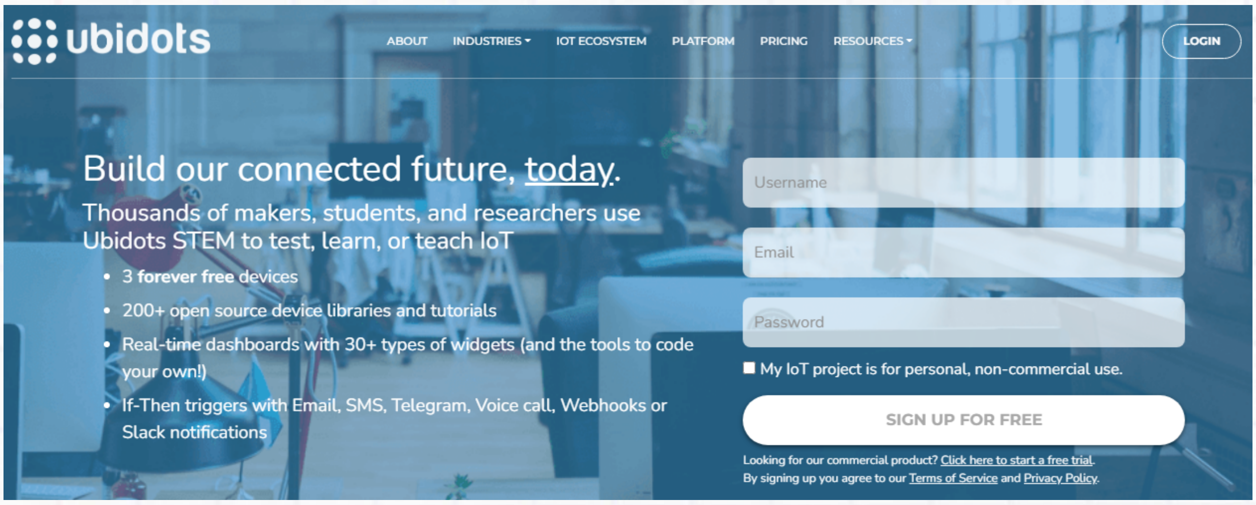
```
If input_device equals Flow_sensor:
        Increment pulse_count and cumulative_count variables by 1.

If input_device equals Reset_button:
        Set pulse_count and cumulative_count variables to 0.
        Set memory_reset_flag variable to 1.
```

The code can now be uploaded to the Dexter board by hitting 'Run'. Note that the project will not work yet as it is incomplete. It will only work once the code for the ESP32 is written and the Ubidots platform is set up.

ESP32 & Ubidots Setup

Downloads & Installation

1) Download and install the Arduino IDE from here.
2) From the IoT Projects page on the Build Club website, download the **ESP32_Flow_Meter** ZIP file and extract it.
3) For monitoring the water flow readings, we will be using the Ubidots IoT Platform. Go to Ubidots.com/stem/ and sign up for free.



Build our connected future, today.

Thousands of makers, students, and researchers use Ubidots STEM to test, learn, or teach IoT

- 3 forever free devices
- 200+ open source device libraries and tutorials
- Real-time dashboards with 30+ types of widgets (and the tools to code your own!)
- If-Then triggers with Email, SMS, Telegram, Voice call, Webhooks or Slack notifications

My IoT project is for personal, non-commercial use.

Looking for our commercial product? Click here to start a free trial.
By signing up you agree to our Terms of Service and Privacy Policy.

Steps

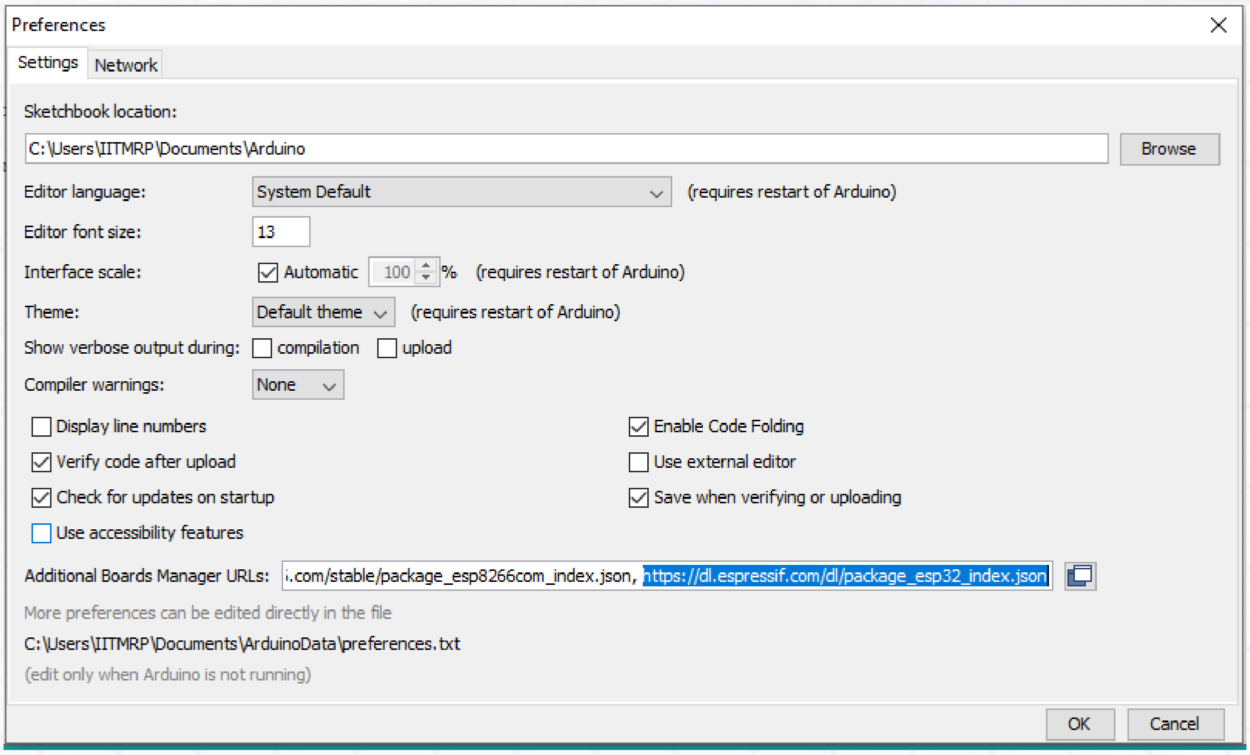
1) Inside the extracted **ESP32_Flow_Meter** folder, you will find 2 folders named **esp32-mqtt-main** & **pubsubclient-master**. Copy them and paste them inside the 'Libraries' folder located in Documents > Arduino > Libraries.

2) Launch the Arduino IDE. Go to **File** > **Open** and select the **ESP32_Flow_Meter.ino** file from the **ESP32_Flow_Meter** folder. The code will now open in the Arduino IDE.

3) Next, go to **File** > **Preferences**. In '**Additional Boards Manager URLs**', if the textbox is empty, paste the following link:

   https://dl.espressif.com/dl/package_esp32_index.json

   If the textbox already has some other links, put a comma after them and paste the above link.



Sketchbook location: C:\Users\IITMRP\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 13

Interface scale: ✓ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

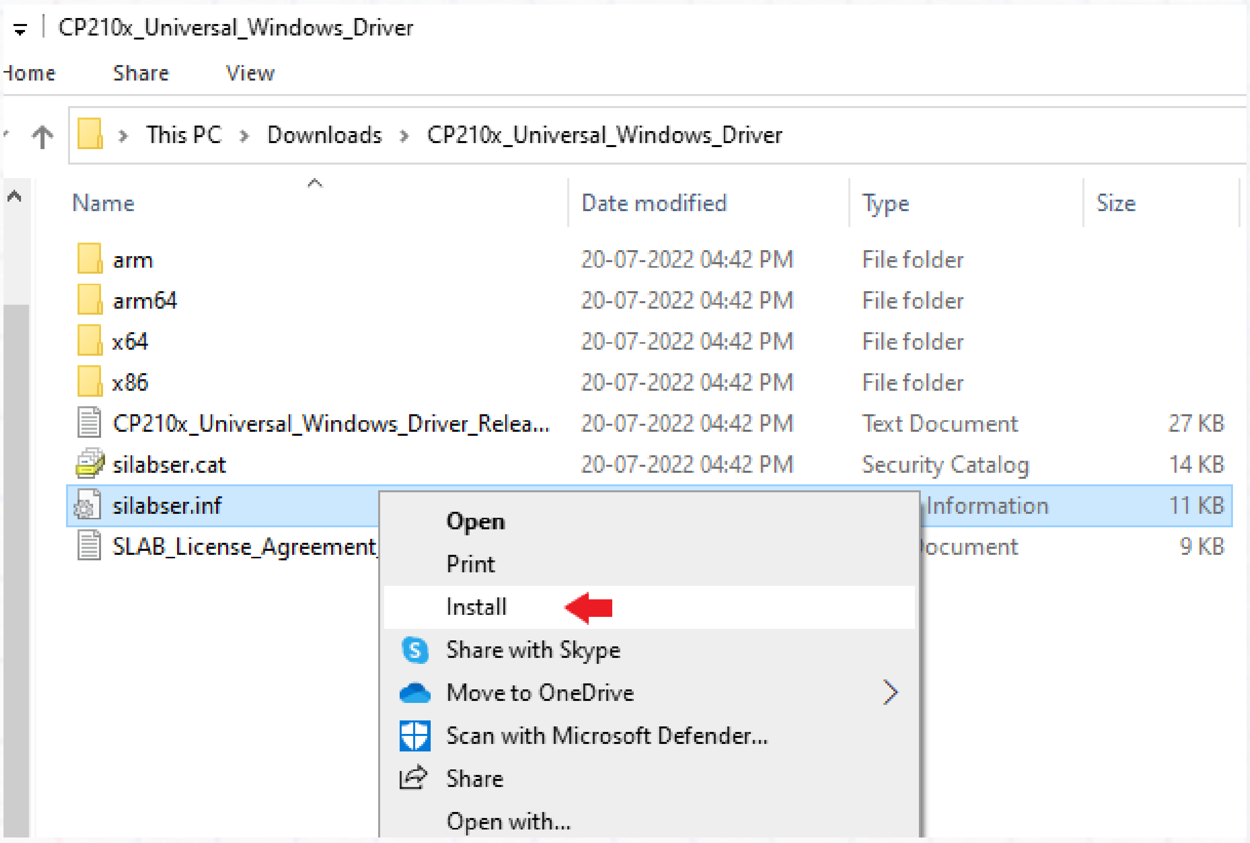
☐ Display line numbers
✓ Verify code after upload
✓ Check for updates on startup
☐ Use accessibility features
✓ Enable Code Folding
☐ Use external editor
✓ Save when verifying or uploading

Additional Boards Manager URLs: i.com/stable/package_esp8266com_index.json, https://dl.espressif.com/dl/package_esp32_index.json

More preferences can be edited directly in the file
C:\Users\IITMRP\Documents\ArduinoData\preferences.txt
(edit only when Arduino is not running)

4) Next, go to **Tools** > **Board** > **Boards Manager**. In the search bar on the top, search 'esp32' and install the package. This will take some time to install.

5) From this link, download the **CP210x Driver** for your OS (eg. CP210x Universal Windows Driver). Extract the ZIP folder. Right-click the **silabser.inf** file inside and select install. Follow the prompts until installation is successful. Restart your PC once installation completes.

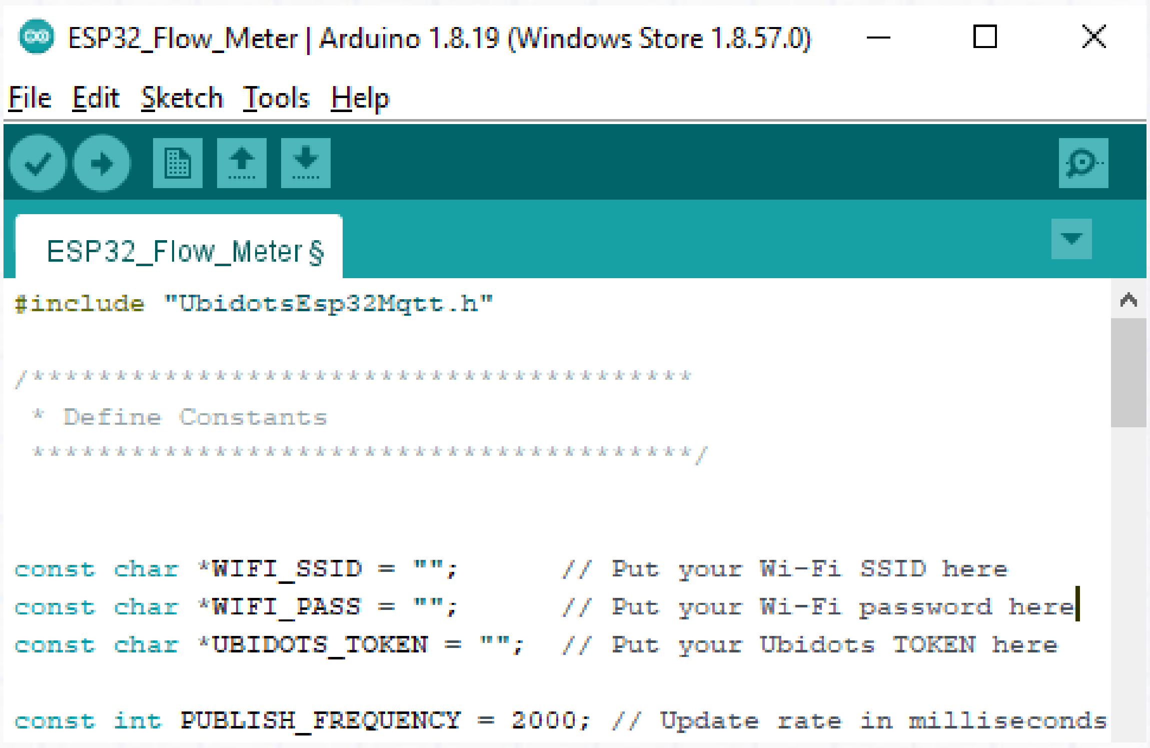

CP210x_Universal_Windows_Driver
Home
Share
View
This PC > Downloads > CP210x_Universal_Windows_Driver
Open
Print
Install
Share with Skype
Move to OneDrive
Scan with Microsoft Defender...
Share
Open with...

Name	Date modified	Type	Size
arm	20-07-2022 04:42 PM	File folder	
arm64	20-07-2022 04:42 PM	File folder	
x64	20-07-2022 04:42 PM	File folder	
x86	20-07-2022 04:42 PM	File folder	
CP210x_Universal_Windows_Driver_Relea...	20-07-2022 04:42 PM	Text Document	27 KB
silabser.cat	20-07-2022 04:42 PM	Security Catalog	14 KB
silabser.inf		Information	11 KB
SLAB_License_Agreement		ocument	9 KB

6) Use 1 USB cable to connect the ESP32 device to your computer and the other to connect the Dexter to the computer for uploading code. Before uploading code to ESP32 from your computer, disconnect the jumper wire from the PC4 pin of the Dexter. This is necessary for the code to upload onto the ESP32.

Open the **ESP32_Flow_Meter.ino** file again in the Arduino IDE. Go to Tools > Board > ESP32 Arduino and select ESP32 Dev Module. In the same Tools menu, under 'Port', select the available COM port (eg. COM10).

7) Now go back to the Flow Meter code in the Arduino IDE. Fill in your wifi connection's name and password within the empty quotation marks next to the WIFI_SSID and WIFI_PASS variables. Ensure the Wifi network has no firewalls, else the ESP32 won't connect to it.

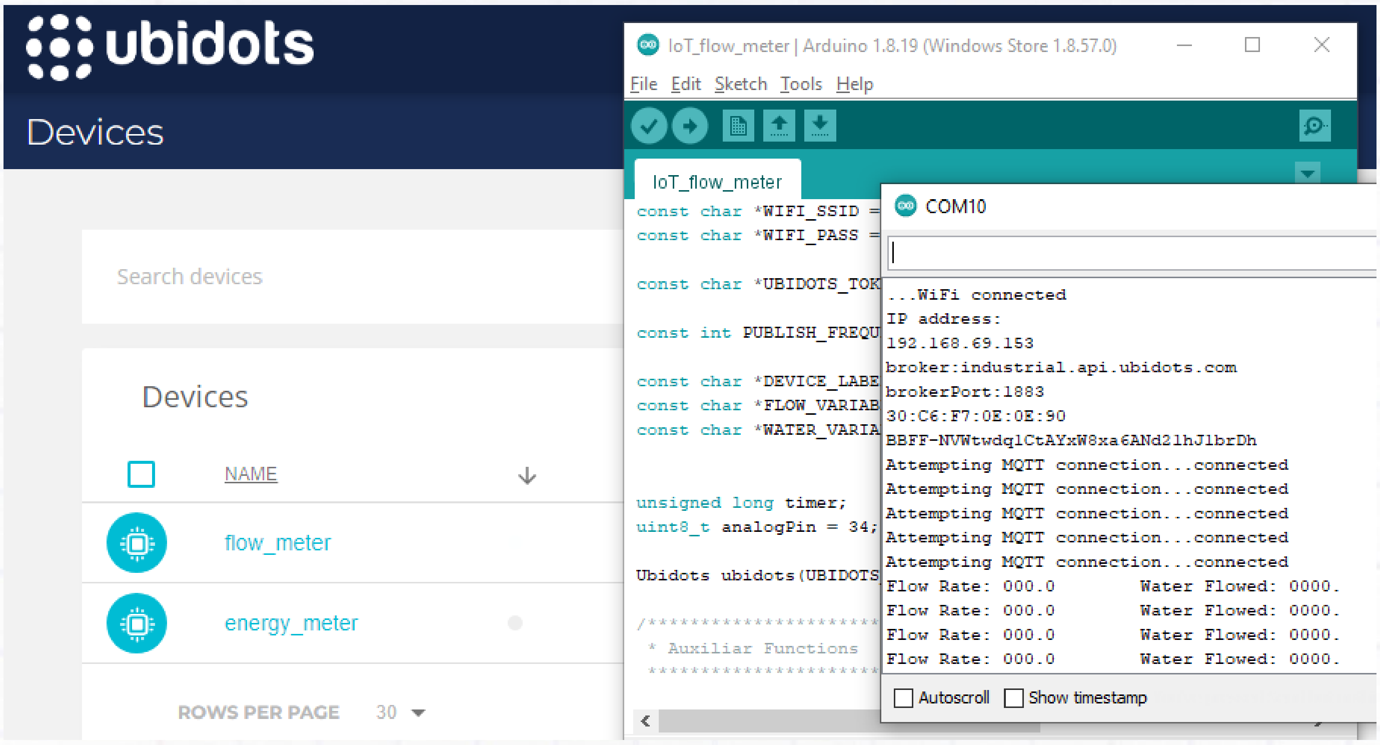

ESP32_Flow_Meter | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
ESP32_Flow_Meter §

```
#include "UbidotsEsp32Mqtt.h"

/****************************************
 * Define Constants
 ****************************************/


const char *WIFI_SSID = "";       // Put your Wi-Fi SSID here
const char *WIFI_PASS = "";       // Put your Wi-Fi password here
const char *UBIDOTS_TOKEN = "";   // Put your Ubidots TOKEN here

const int PUBLISH_FREQUENCY = 2000; // Update rate in milliseconds
```

8) On the Ubidots website, click on your profile icon and select API credentials. Copy the 'Default token' on the top right part of the page and paste it into the UBIDOTS_TOKEN variable, which is immediately below the Wifi variables in the Arduino code.

9) Click the 'Verify' button on the Arduino IDE - shown by a tick symbol. Once you receive the "Done compiling" message, click the 'Upload' button, shown as a right arrow near the 'Verify' button. Wait until the "Done uploading" message appears.

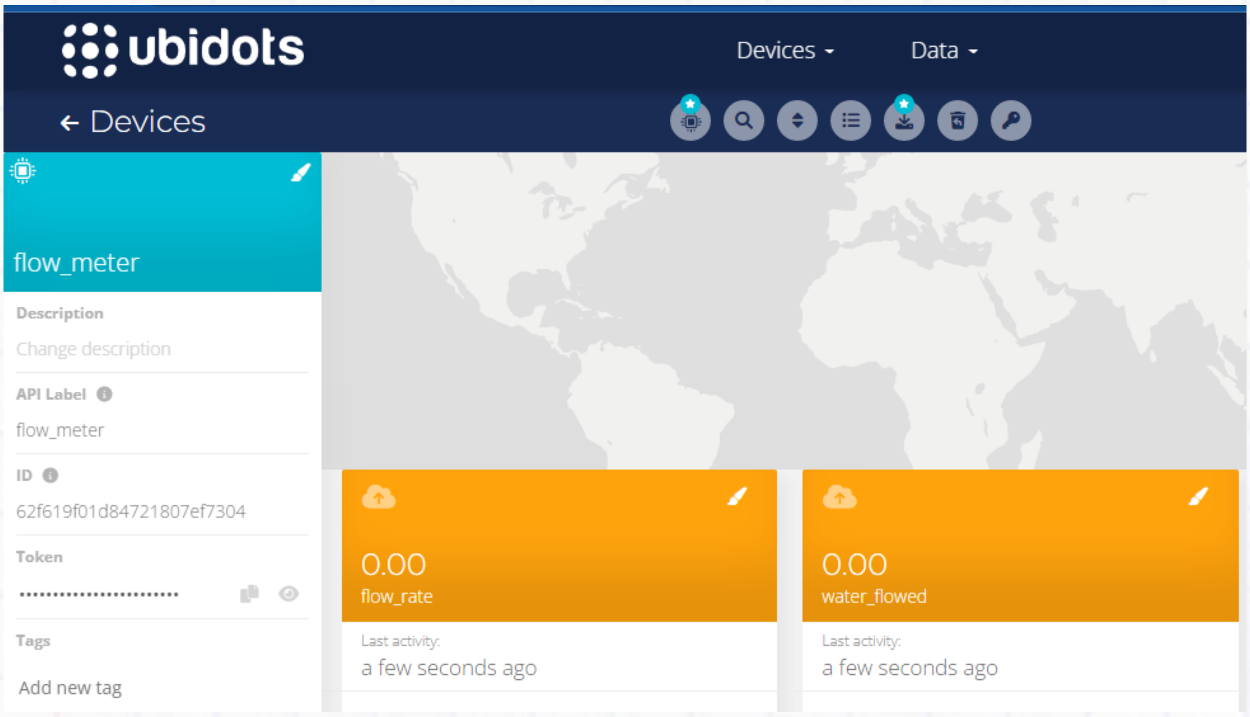
10) Now reconnect the jumper to the PC4 pin as before. Click the magnifying glass icon on the top-right of the Arduino IDE. It will open the Serial Monitor. If the connection was successful, the Serial Monitor should look like:



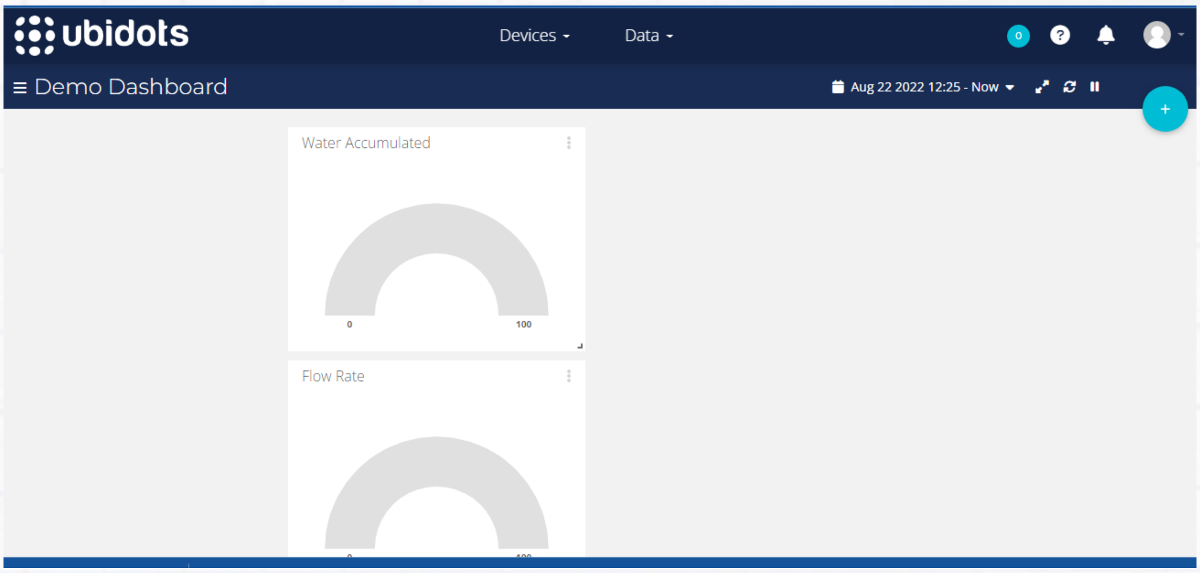
```
const char *WIFI_SSID =
const char *WIFI_PASS =

const char *UBIDOTS_TOK

const int PUBLISH_FREQU

const char *DEVICE_LABE
const char *FLOW_VARIAB
const char *WATER_VARIA


unsigned long timer;
uint8_t analogPin = 34;

Ubidots ubidots(UBIDOTS

/*********************
 * Auxiliar Functions
 *********************
```

```
...WiFi connected
IP address:
192.168.69.153
broker:industrial.api.ubidots.com
brokerPort:1883
30:C6:F7:0E:0E:90
BBFF-NVWtwdqlCtAYxW8xa6ANd2lhJlbrDh
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Flow Rate: 000.0        Water Flowed: 0000.
Flow Rate: 000.0        Water Flowed: 0000.
Flow Rate: 000.0        Water Flowed: 0000.
Flow Rate: 000.0        Water Flowed: 0000.
```

11) Now go to the Devices page of the Ubidots website. You will find a device named **flow_meter** containing variables **flow_rate** and **water_flowed**.



12) Go to Data > Dashboards and click the **+** symbol. Select 'Gauge' and name it 'Flow Rate'. Click 'Add Variables' and select the flow_rate variable under the flow_meter device. Create another gauge similarly for the water_flowed variable.



Tasks

1) The Flow Rate value in the Serial Monitor should read zero. If the Water Flowed value isn't initially zero, press the reset push button on the breadboard. Now slowly blow into the water meter pipe and check whether the *flow_rate* and *water_flowed* variables change in the Serial Monitor. Go to the Ubidots Dashboard page - the meter gauges should reflect the change in values.

2) Press the push button to reset the readings. In the STM32 IDE, open Live Expressions (Taught in the Maze Robot project) and add the *cumulative_count* variable to it. Debug the program, similar to how we had done for the sensors in the Maze Robot project.

   Now hold the Flow meter pipe over a bucket and pour exactly 200ml of water. Check the reading in Ubidots - is it exactly 200 ml? If not accurate, you will have to calibrate it as follows:

   Calibration: After 200 ml of water is poured, check the value of the *cumulative_count* variable in Live Expressions. In **app.c**, you will find a variable called *mL_per_count* - which is a fraction with 200 as the numerator. Replace its denominator value to the value of the *cumulative_count* variable in Live Expressions.

3) Now that calibration is done, we can test our IoT Water Flow Meter. Reset the readings using the push button and pour 450ml of water slowly. On the Ubidots dashboard, you should be able to view the flow rate and water accumulated values live. Once flow is completed, the value of *water_flowed* should become 450 ml. Now pour a further 320 ml of water. The *water_flowed* value will increase to 770 ml. Pressing the push button will now reset it back to 0 ml.

Real-world Application

You have now learnt how to build an IoT based Water Flow Meter and are ready to apply it in real-world scenarios!

This Water Meter can be used in several applications. Identify locations in your college or home where it can be installed - this could be a hand wash area, a distribution pipe of a water tank, or even a drinking water dispenser.

One very simple and practical use-case is in an RO water purifier. Have you ever wondered how an RO water purifier works? Using Reverse Osmosis technology, it concentrates impurities and sediments in one portion of the water - which is wasted - leaving the purified portion for drinking. But do you have any idea how much water is wasted?

Taking permission from your college authorities, locate an RO water purifier in your college and install the Water Meter in the pipe that drains out the waste water. It must be installed in such a way that the electronics are protected from water, heat or physical damage. Make sure that there is access to Wifi in that area. Track the wastage quantities over different periods of time - a day, a week, a month... - and note these values.

Additionally, you can build one more Water Meter or use the same to measure the amount of purified water consumed form the same RO system. You will be surprised to know the amount of water wasted to purify just 1L of water!

Make a video presenting the installation of the water meter and the results/insights you got about the RO system. Share it with the Build Club Community on the Discord Server! Also share your own unique applications of the IoT Water Meter!