Redback Operations

IoT Sensor and Development Team &
Data Science Team

By
Shashvat Joshi (Data Analysis – Algo/Prototype Team Lead)

Introduction

Cadence in cycling is defined as the number of revolutions the pedal makes per minute (RPM) as we ride. Put in a layman term, it is rate at which the cyclist pedals' (RPM) is directly proportional to the wheel speed. A skilled cyclist's legs move in a smooth and coordinated pattern on any kind of road. So, the ease of cycle comes from apply proper cadence at the correct time.

The rotational speed is proportional to the cadence through a coefficient *g*, represented by a gear ratio,

$$\omega(t) = g(t)c(t)$$

If the speed of the bicycle is single, i.e., with fixed gear ratio, *g(t)* = *g*.

Various sensors that can be used to calculating speed and thus cadence.

Cadence Sensor

Some commercially available popular cadence sensors for indoor cycling provide the cadence in real-time. Some commercially available sensors are:

1. Moofit Cadence
2. Wahoo
3. Magene
4. Garmin
5. Other cadence sensors that broadcast Speed & Cadence specification

These sensors typically operate on Bluetooth in GATT (Generitt ATTribute Profile) mode. This defines the commands and data that can be exchanged between bike sensor devices and the client device such as a phone, tablet, or BLE capable microcontroller. These sensors directly provide

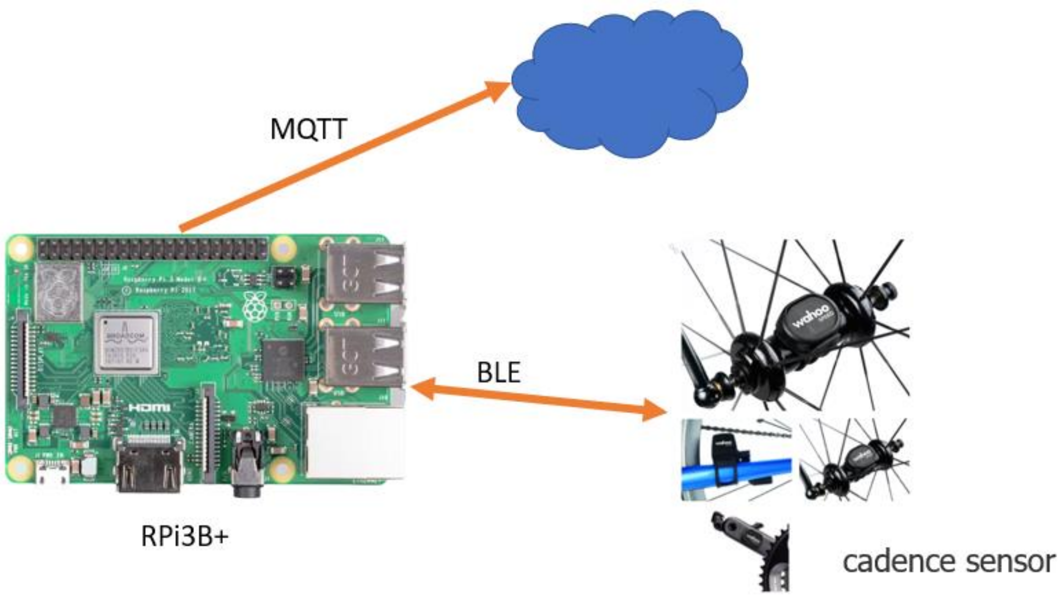
- Wheel revolutions - calculate speed from wheel diameter
- Crank revolutions – gives pedaling cadence

Note: Not all Cycling sensor support all the above characteristics. Some individual sensors will report just speed or just cadence.

The cadence/speed sensor typically reads speed and cadence revolutions based on a pair of magnets affixed to a spoke and crank

We can use a Raspberry Pi 3 Model B+ (RPi3B+), which is a single-board computer with both dual-band wireless LAN and Bluetooth 4.2/BLE.

The block Diagram is as below:

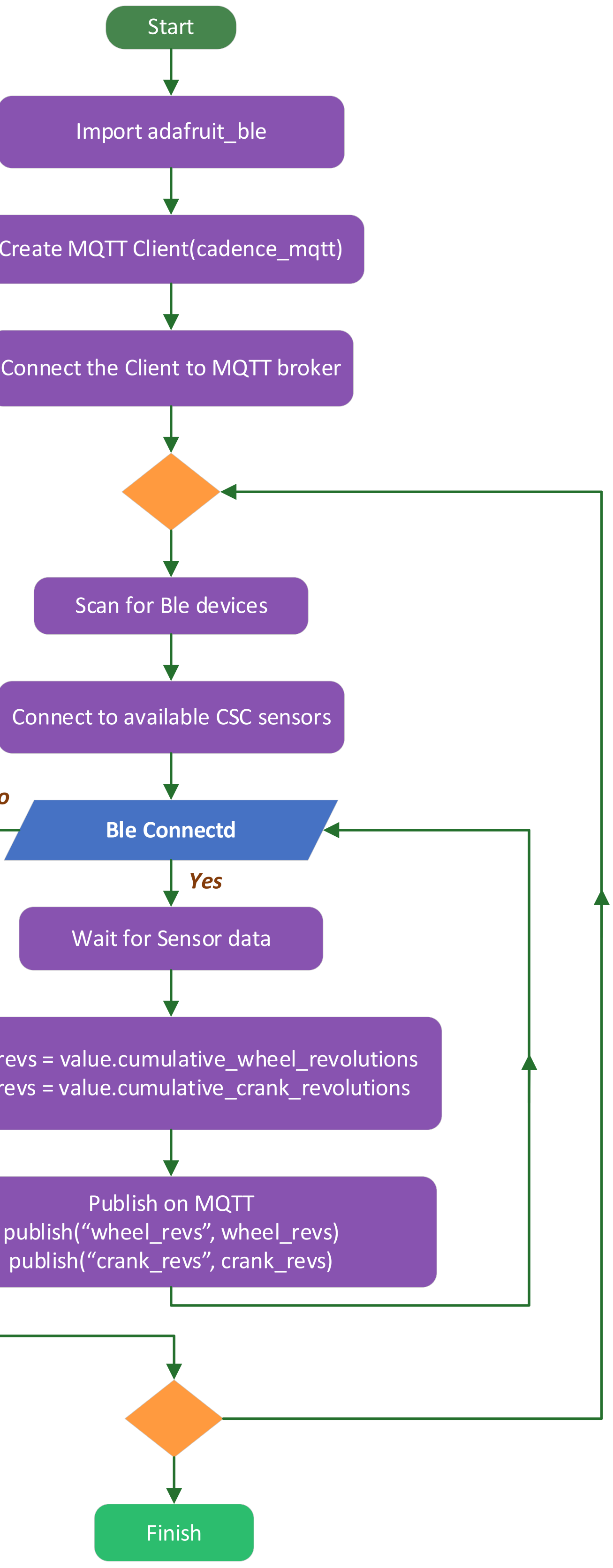


We first install the cadence driver

```
pip3 install adafruit-circuitpython-ble-cycling-speed-and-cadence
```

Flow chart - 1

The flow chart of the algorithm is as below

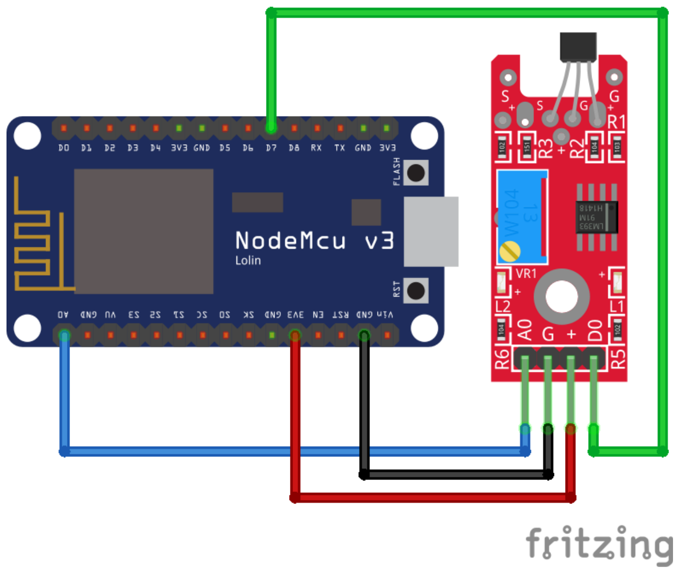

Start
Import adafruit_ble
Create MQTT Client(cadence_mqtt)
Connect the Client to MQTT broker
Scan for Ble devices
Connect to available CSC sensors
No
Ble Connectd
Yes
Wait for Sensor data
wheel_revs = value.cumulative_wheel_revolutions
crank_revs = value.cumulative_crank_revolutions
Publish on MQTT
publish("wheel_revs", wheel_revs)
publish("crank_revs", crank_revs)
Finish

Hall Sensor

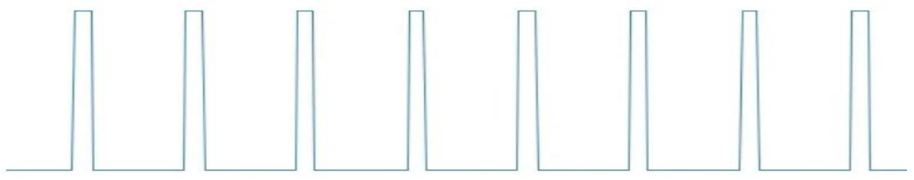
Another simpler method is to use a hall sensor to measure the RPM. Let us use the sensor KY-024. The digital value of the KY-024 hall sensor module change between 0 and 1.

- 0 for a magnet is far from the sensor (magnetic field too weak)
- 1 if the magnet is near the sensor (magnet field strong)

The circuit can be as below when we use the NodeMCU module to communicate on Wi-Fi to MQTT server


D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
NodeMcu v3
Lolin
FLASH
RST
A0 GND VU S3 S2 S1 SC SO SK GND 3V3 EN RST GND Vin
S G R1 R2 R3 VR1 L1 L2 A0 G + D0 R5 R6
fritzing

A magnet is placed typically on the spoke. To increase the accuracy, we can increase the number of magnets uniformly spaced. We mount the hall sensor on the frame in such a way that the magnet/s pass over the sensor. When the cycle starts spinning, it will pass next to the sensor and trigger it, triggering the interrupt in the code and we can get the rpm count.



To communicate with the MQTT broker, we need to install the PubSubClientlibrary. We can download from below link

Link: http://osoyoo.com/wp-content/uploads/samplecode/pubsubclient.zip

We Unzip above file and move the unzipped folder to Arduino IDE library folder.

To initialize the Wi-Fi and MQTT, we follow the following steps.

1. Include the Wi-Fi and MQTT libraries in the scratch code

```
#include <ESP8XXXWiFi.h>
#include <PubSubClient.h>
```

2. Hotspot Configuration:

```
const char* ssid = “your_hotspot_ssid”;
const char* password = “your_hotspot_password”;
```

put appropriate ssid and password on there.

3. Start Wi-Fi

```
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
```

4. MQTT Server Address Setting

```
const char* mqtt_broker = “broker.xyz.com”;
```

5. Connect to MQTT Client

```
WiFiClient espClient;
PubSubClient client(espClient);
client.setServer(mqtt_broker, mqtt_port); // mqtt_port 1883
client.connect(clientId) // Any String that is unique to the
client
```

6. To publish on MQTT once connected

```
client.publish("outTopic", "data-xyz");
```

Publish on specific topic for cadence/rpm on the server with appropriate rpm data.

7. To calculate rpm, we measure the pulses generated by the hall sensor. To do this we install an interrupt that increments a counter when it is triggered. We also store the millis() every time the hall sensor increments.

In the main loop:

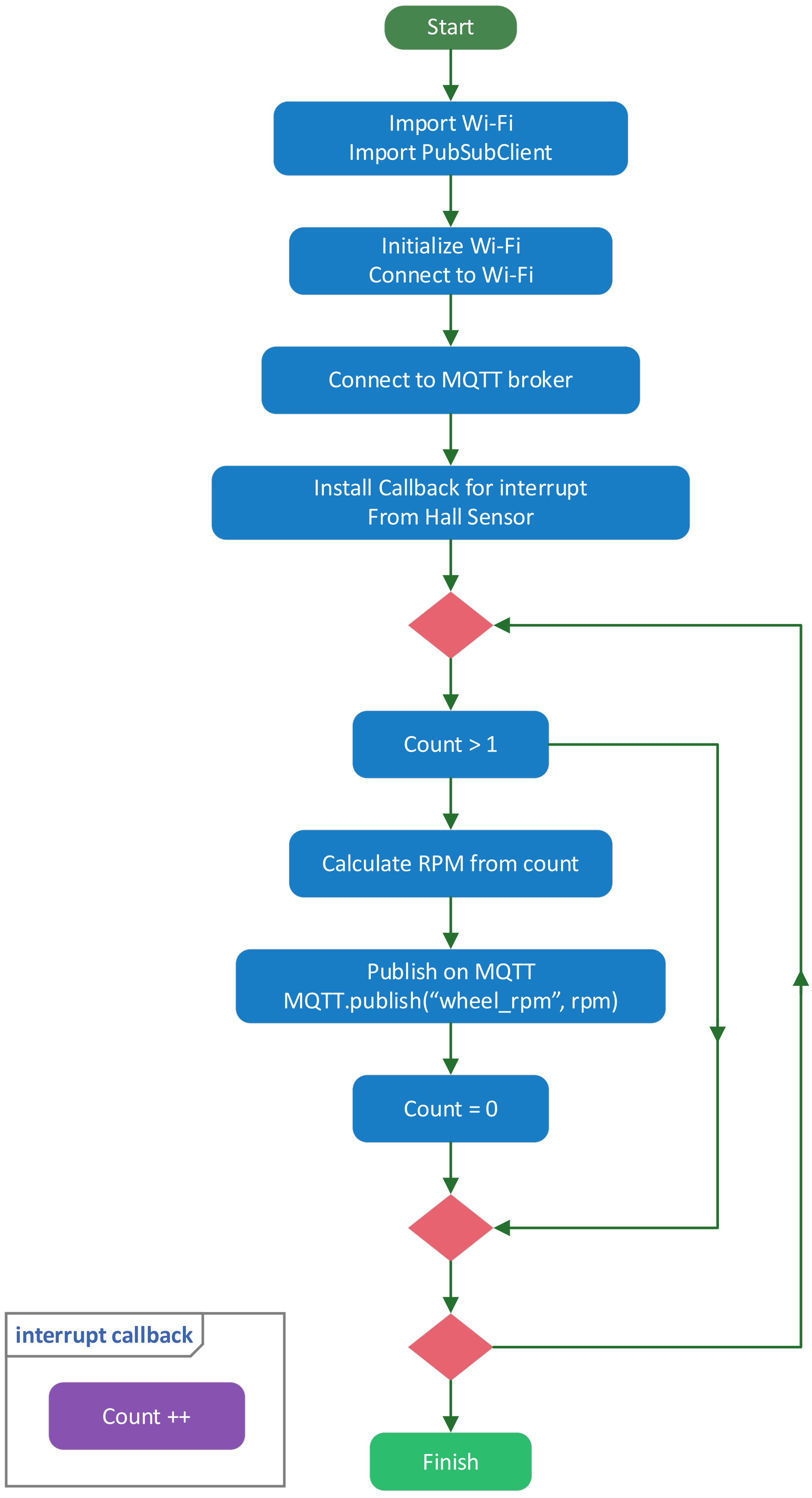
```
if (pulses>=1) {
    rpm = 60000.0/(millis()-timeold)*pulses;
    rps=rpm/60.0;
    timeold = millis();
    pulses=0;
}
```

In the interrupt:

```
//This Sub-program runs when interrupt is triggered
void counter() {
   pulses++; //every time this sub-program runs, the
number of pulses is increased by 1
}
```

Flow chart - 2

Flow Chart for RPM using Hall Sensor

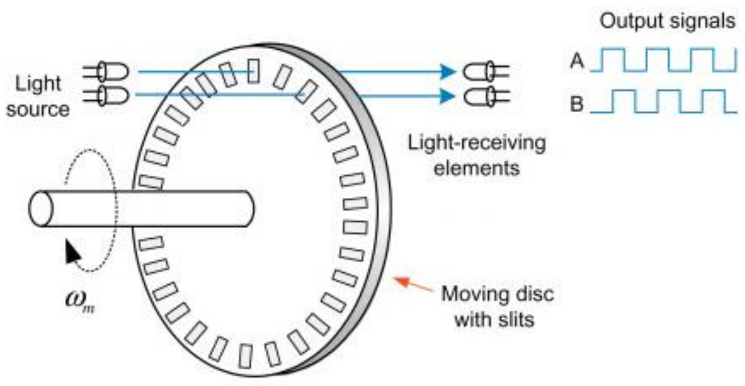


Rotary Encoders

Rotary encoder modules can be used for calculating the rotation speed. A rotary encoder is a device that translates rotational motion into electrical impulses. There are two output pins on the rotary encoder: A and B. Electrical signals will be generated on pins A and B if the rotary encoder is properly mounted and rotates with the motor. The rotation speed, direction of rotation, and position of the motor can all be determined by monitoring these signals.

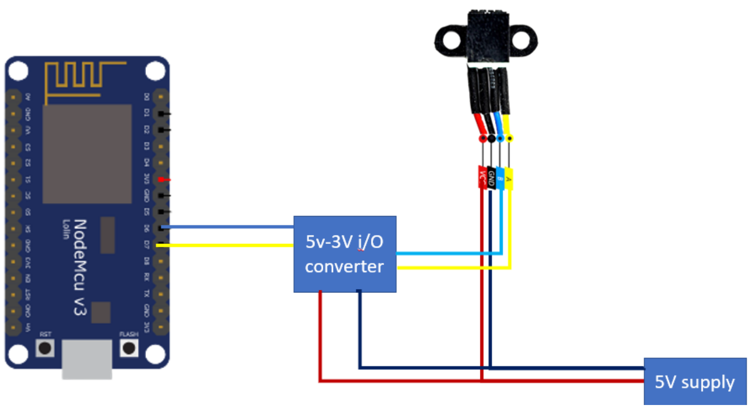
Working Process

Two discs are positioned across each other in the photoelectric encoder module. On one disc, there is a light source, and on the other disc, there are two light-receiving elements (photoresistors). A moving disc with uniformly spaced slots sits between these two discs. The light receivers A and B are alternately connected and disengaged by spinning this movable disc. As a result, the outputs will emit a square wave. You may calculate the amount of rotation by counting these pulses. Furthermore, because the two outputs A and B have a 90-degree phase difference, we may determine the direction of rotation by determining which output is ahead of the other.



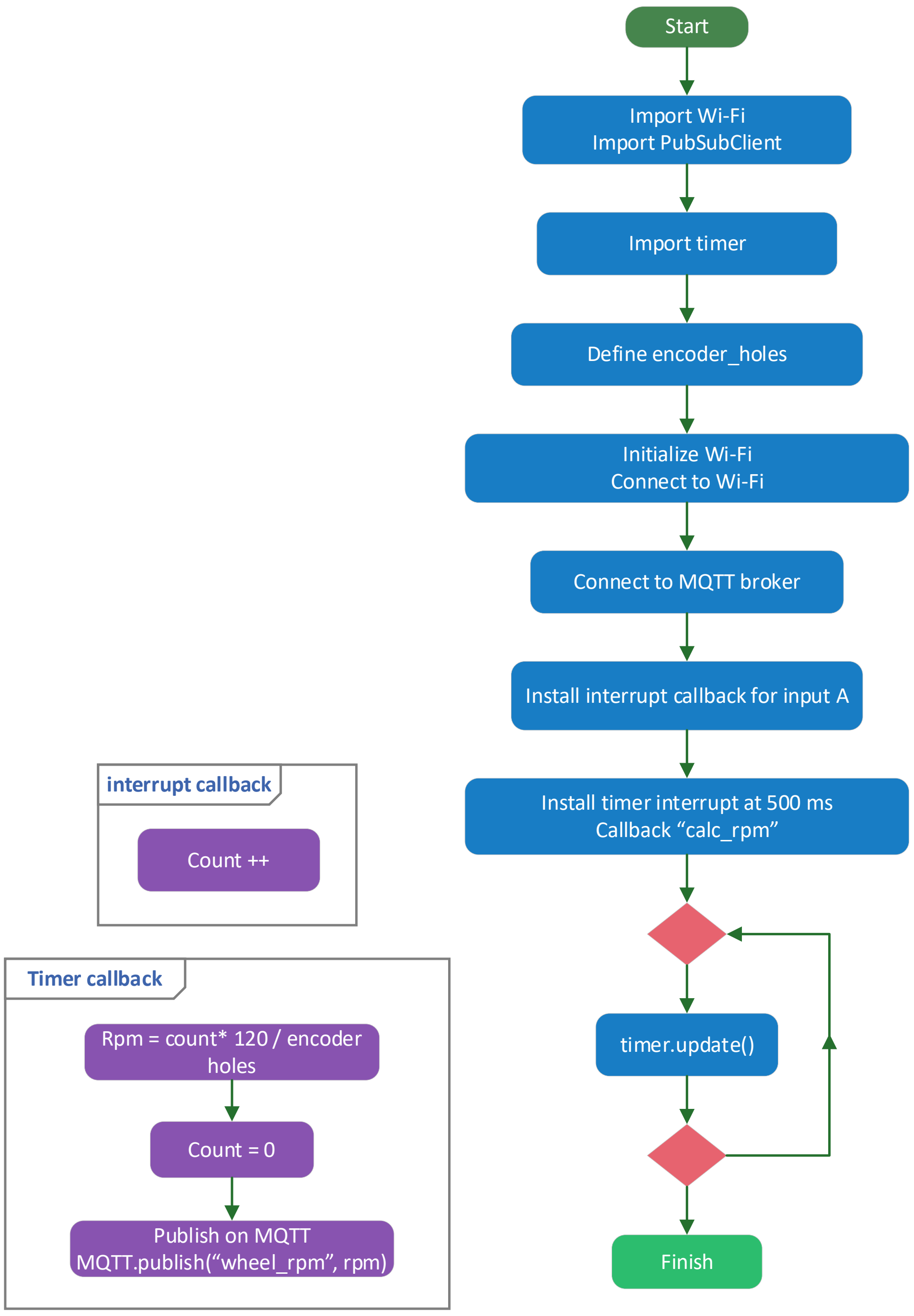
Circuit

We then put a level converter for the encoder outputs A and B as they are all at 5V and the Node MCU accepts 3.3V



Flow chart - 3

The flow chart/pseudo code how rpm can be generated with better accuracy in case of encoder as the accuracy depends on the number of encoder holes.



Conclusion:

As we have seen we can use various kinds of sensors to measure cadence/rpm of the wheel as well as the crank. Some of these sensors can give us processed data but can be costly whereas the others can be simple and cheap but need processing. A high accuracy encoder will provide accurate results and can also give the direction of cadence.

We can also use dual sensor attached to the wheel and the crank to measure cadence and rpm separately or we can use various other algorithms to predict the cadence from the rpm. talks real-time cadence estimation using the wheel speed (rpm) specifically in the presence of gears.