**REPORT ON WINTER INTERNSHIP 2019-20**

# Topic: IOT BELT

# Under the Guidance Of :- Dr. Anjali Yeole

**Team Members:**

1. Tanish Sahijwani

2. Priya Gangwani

3. Ganesh Gaonkar

4. Prathmesh Indalkar

5. Pranay Pagare

**REPORT ON WINTER INTERNSHIP 2019-20**

**IOT BELT FOR DOG**

horizontal line

# 

# Abstract-:

# There is an increasing number of issues regarding various animal health condition and movements. And in recent era, animals have become an integral part of a human life. And hence, an animal health monitoring and tracking system using UBIDOTS is developed. UBIDOTS is more and more adopted in a wide range of applicative scenarios. To track the health of an animal, sensors such as the temperature sensor, pulse rate sensor and the motion sensor, GPS are used. With the advancement in technology and the existence of internet, we practically can connect any device to internet and implement the concept of IOT.

# Introduction-:

# As the life of humans are important, similarly the life of pets are also important. They cannot tell about the problems that they face, but by observing their body movement and physical behaviour we can identify their problems. As nowadays, people don't get time from work to take care of their pet so, this device helps to get the information about the pet’s health.

# Internet of Things (IoT) provides efficient technical support for the biology and zoology research. We can use this technology for getting information about the pets. The sensors such as temperature sensor, GPS, pulse sensor, motion sensor helps to monitor the dog. These sensors are interfaced with the NodeMCU ESP8266. NodeMCU is the wifi module which collects the information from the sensors and displays it on the website. These all temperature sensors are located on the belt of dog. Sensors provide the body temperature, pulse rate (in bpm), location and activity or motion of the dog.

# The information is displayed on Ubidots. Ubidots is an Internet of Things (IoT) data analytics and visualization company which converts the sensor data into information. This helps the user to know about the health of dog from any corner of the world by just logging in into ubidots.

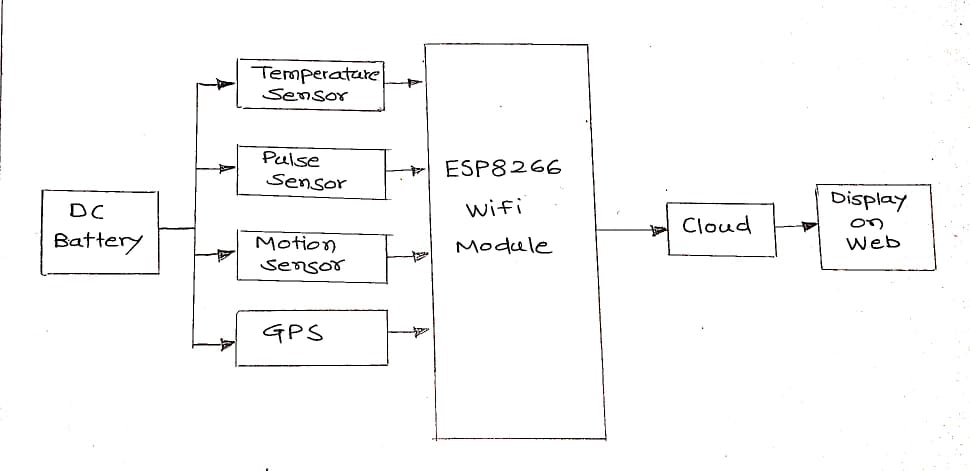
## Hardware Requirement-:

|  |  |  |
| --- | --- | --- |
| Sr no. | Sensor | Sensor Name |
| 1 | Temperature Sensor | DS18B20 Waterproof Temperature Sensor Probe |
| 2 | Pulse Sensor | Pulse Sensor |
| 3 | Motion Sensor | Triple Axis Accelerometer + Triple Axis Gyroscope - MPU6050, GY-521 |
| 4 | GPS Module | Ublox NEO-6M GPS Module |
| 5 | WIFI-Module | NodeMCU ESP8266 CP2102 WIFI Module Node MCU |
| 6 | Rechargeable Battery | Lipo Battery Model KP-502530 1000mA |
| 7 | Battery Charger | TP4056 1A Li-Ion Battery Charging Board Micro USB with Current Protection |
| 8 | Connecting Wires | Male to male Male To female |
| 10 | Black Cotton Cloth | 1.5 Meter |

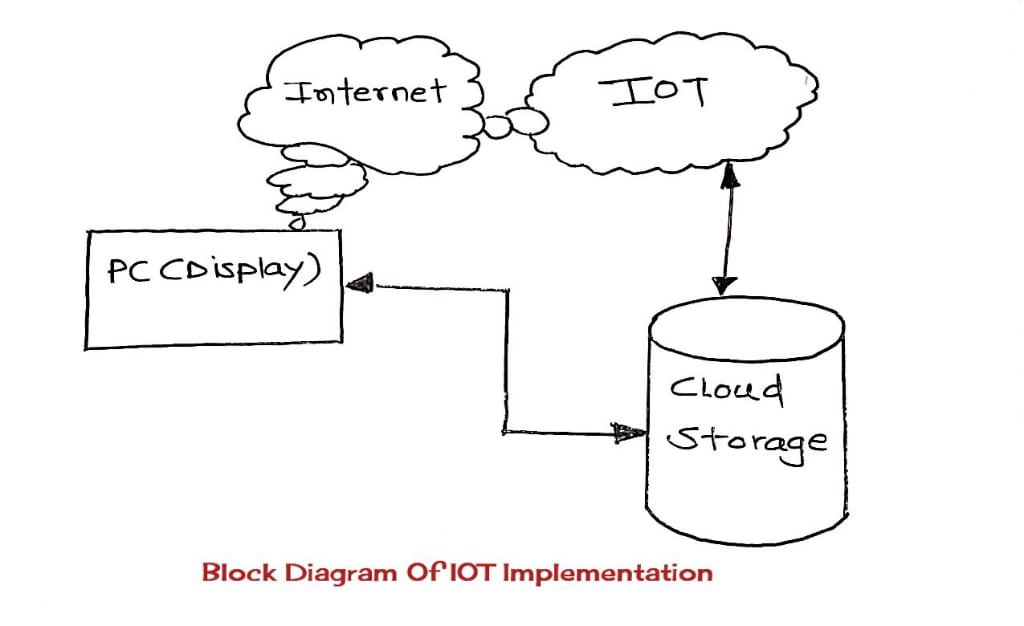
## Software Requirement-:

1. Arduino IDE Version
2. UBIDOTS

* **Block Diagram Of Our Project-:**

****

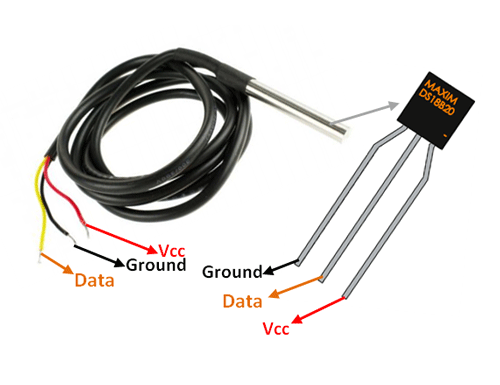
* **Block Diagram Of IOT Implementation-:**

****

The Internet of Things (IoT) is a network of physical objects—devices, vehicles, buildings and other items embedded with electronics, software, sensors, and network connectivity—that enables these objects to collect and exchange data. IoT allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration between the physical world and computer-based systems, and resulting in improved efficiency, accuracy and economic benefit.

* **Components Features**-:

1. **DS18B20 Temperature Sensor-:**

****

* **DS18B20 Sensor Specifications-:**
* Programmable Digital Temperature Sensor
* Communicates using the 1-Wire method
* Operating voltage: 3V to 5V
* Temperature Range: -55°C to +125°C
* Accuracy: ±0.5°C
* Output Resolution: 9-bit to 12-bit (programmable)
* Unique 64-bit address enables multiplexing
* Conversion time: 750ms at 12-bit
* Programmable alarm options
* Available as To-92, SOP and even as a waterproof sensor
* **Pulse Sensor (For Heart Beat Variability)-:**

****

**Pulse Sensor Specifications**

* Biometric Pulse Rate or Heart Rate detecting sensor
* Plug and Play type sensor
* Operating Voltage: +5V or +3.3V
* Current Consumption: 4mA
* Inbuilt Amplification and Noise cancellation circuit.
* Diameter: 0.625”
* Thickness: 0.125” Thick
* **GPS Module NEO-6M-:**

****

**Features of NEO-6M GPS Module:-**

* 5Hz position update rate
* Operating temperature range: -40 TO 85°C UART TTL socket
* EEPROM to save configuration settings
* Rechargeable battery for Backup
* The cold start time of 38 s and Hot start time of 1 s
* Supply voltage: 3.3 V
* Configurable from 4800 Baud to 115200 Baud rates. (default 9600)
* SuperSense ® Indoor GPS: -162 dBm tracking sensitivity
* Support SBAS (WAAS, EGNOS, MSAS, GAGAN)
* Separated 18X18mm GPS antenna

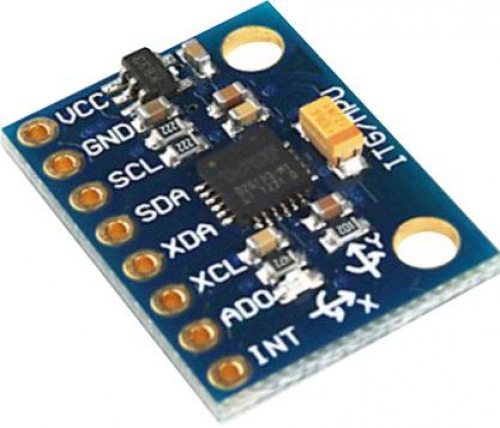
### 

### 

### 

### 

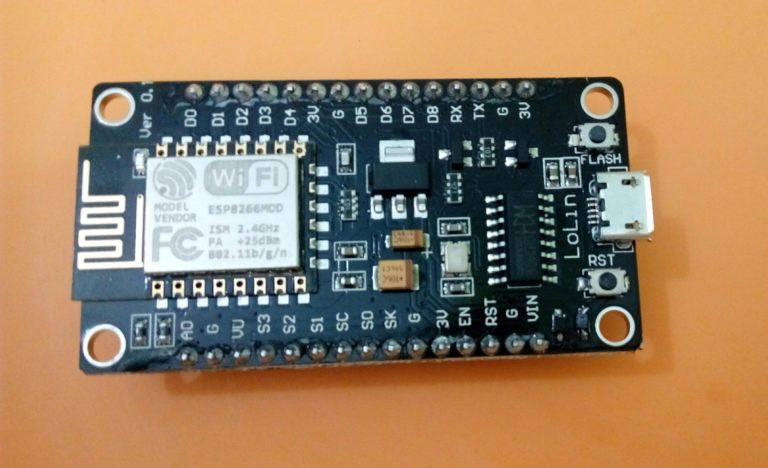
### MPU6050 (Motion Sensor for Activity and Intensity)



### MPU6050 Features-:

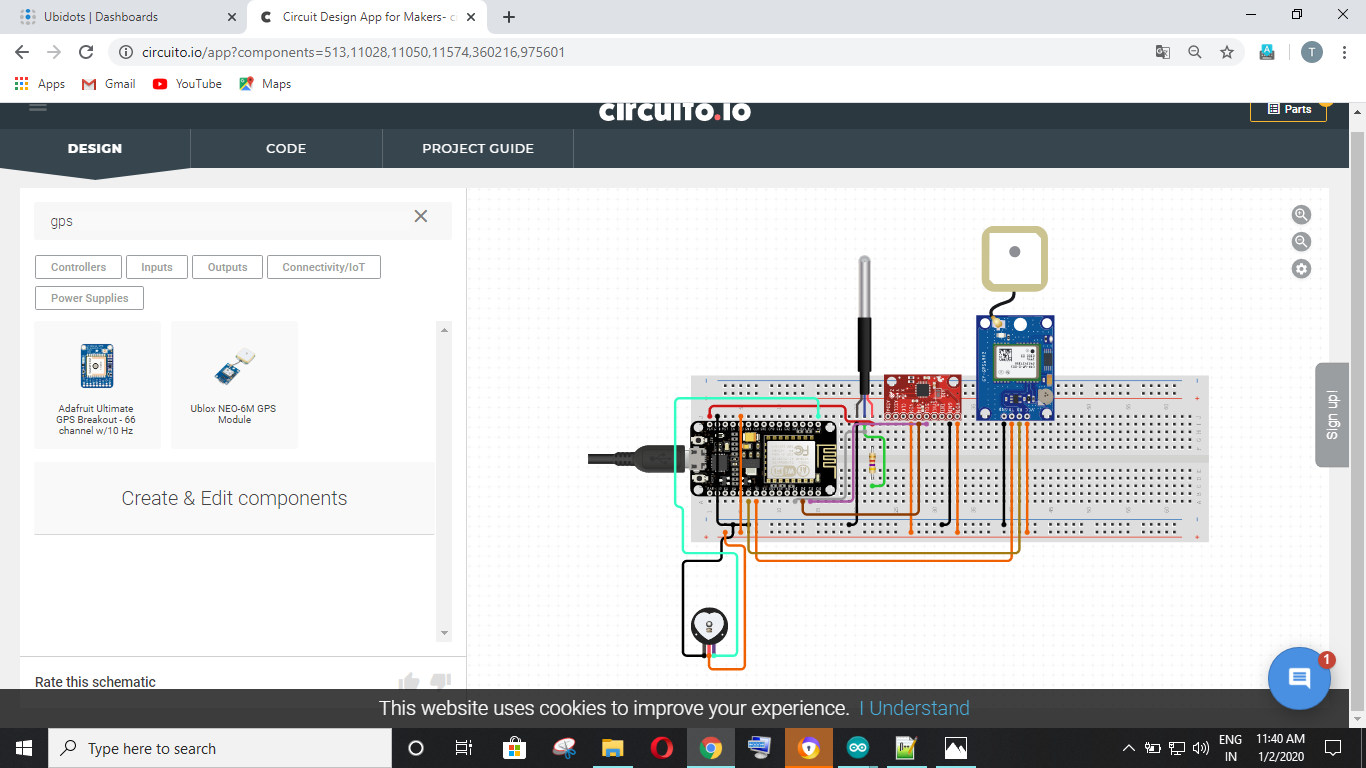
* MEMS 3-axis accelerometer and 3-axis gyroscope values combined
* Power Supply: 3-5V
* Built-in 16-bit ADC provides high accuracy
* Built-in DMP provides high computational power
* Can be used to interface with other IIC devices like the magnetometer.

### ESP8266-01 ( Wifi Module)-:



### ESP8266-01 Features-:

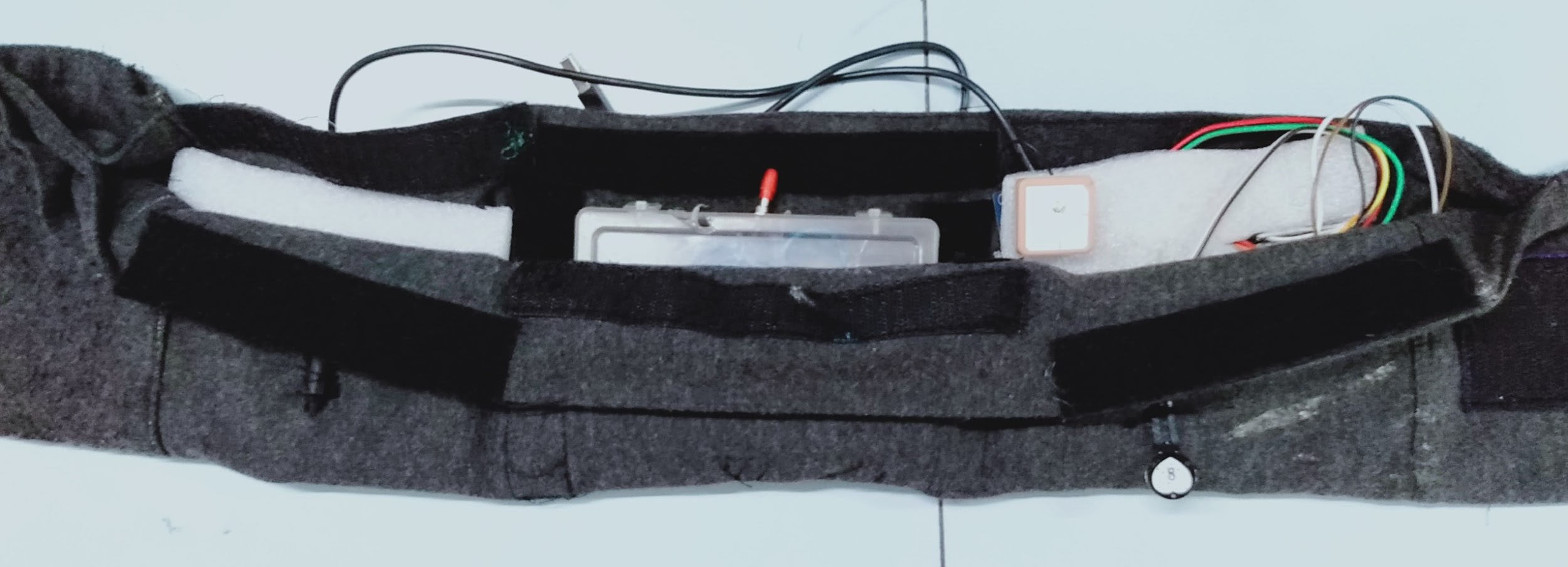
* Low cost, compact and powerful Wi-Fi Module
* Power Supply: +3.3V only
* Current Consumption: 100mA
* I/O Voltage: 3.6V (max)
* I/O source current: 12mA (max)
* Built-in low power 32-bit MCU @ 80MHz
* 512kB Flash Memory
* Can be used as Station or Access Point or both combined
* Supports Deep sleep (<10uA)
* Supports serial communication hence compatible with many development platforms like Arduino.
* **Circuit Diagram-:**



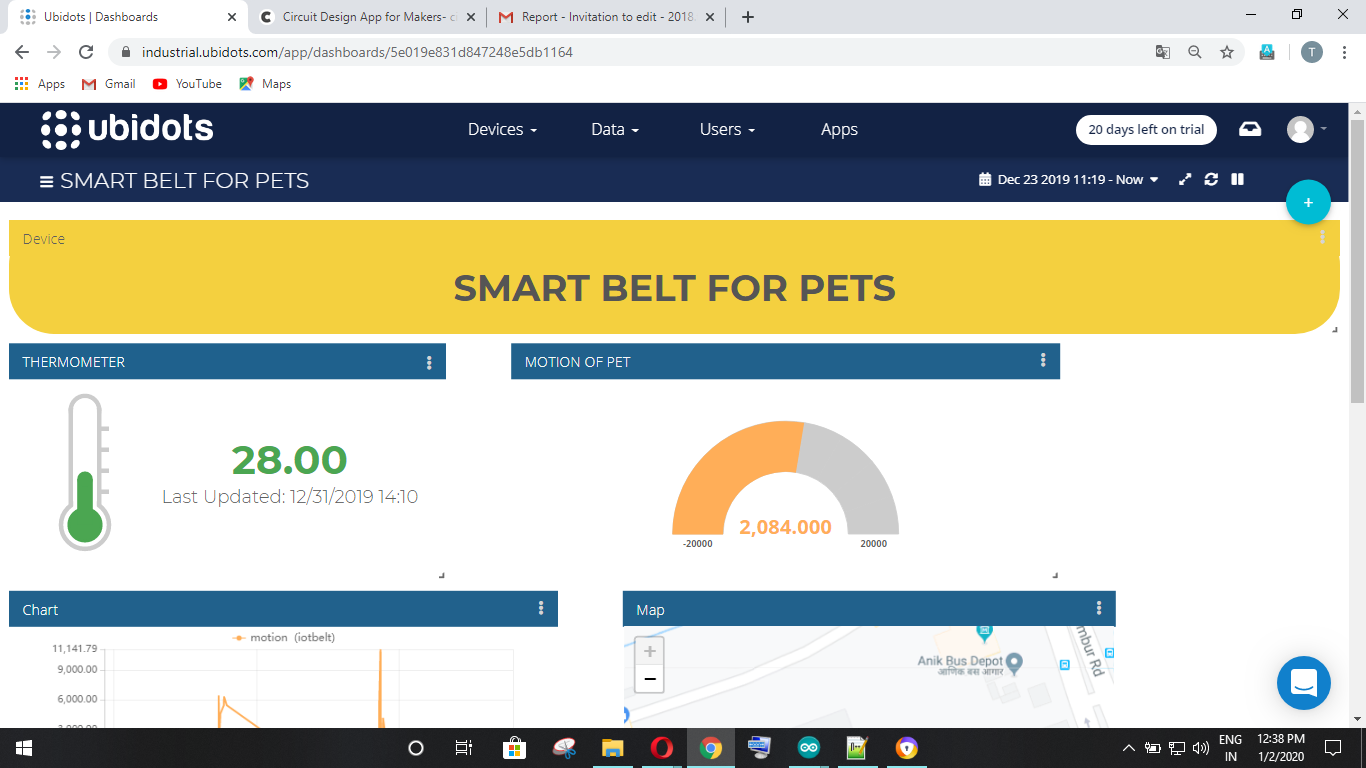
* **Design Of Belt-:**

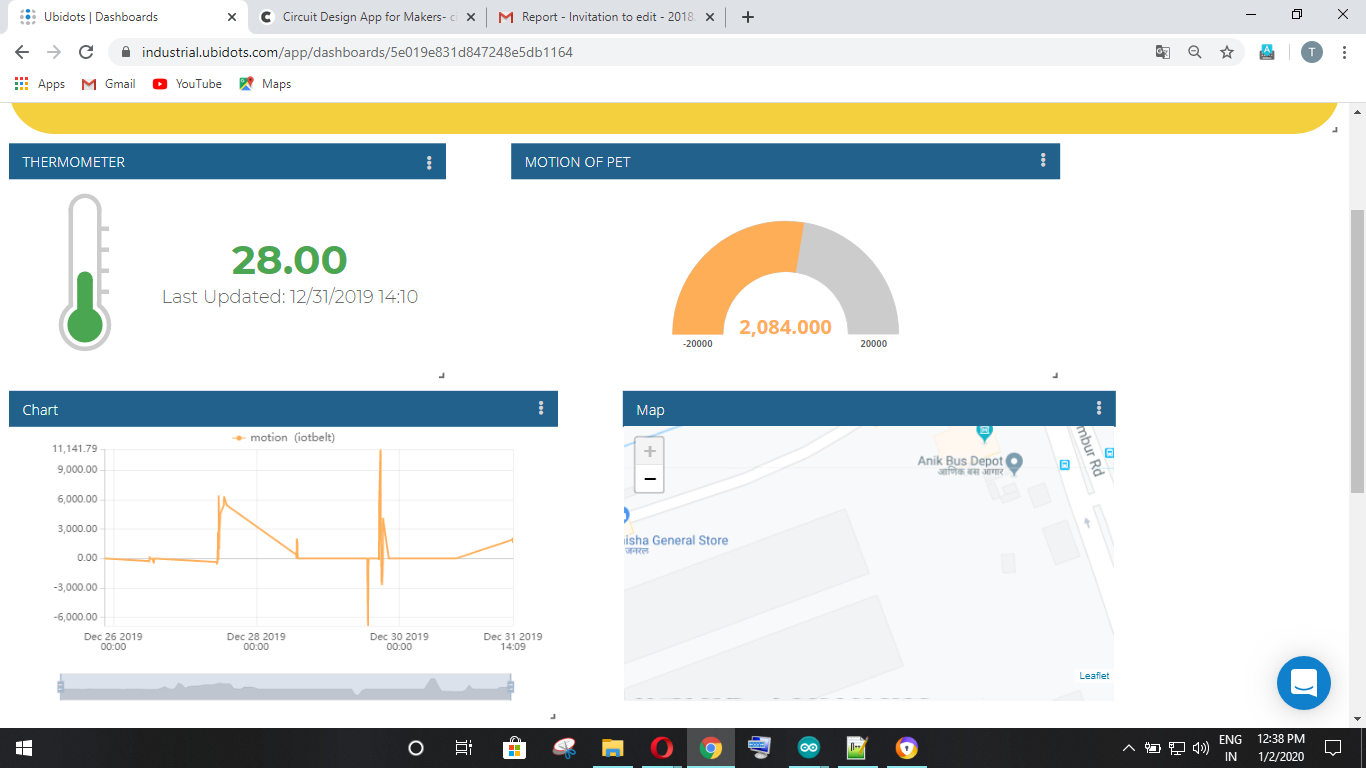
****

* **Internal Structure Of Belt-:**

****

* **Output on UBIDOTS-:**

****

****

**Code-:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Include Libraries

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "Arduino.h"

#include "MPU6050.h"

#include "Wire.h"

#include "I2Cdev.h"

#include <UbidotsESPMQTT.h>

#include "UbidotsESPMQTT.h"

#include <OneWire.h>

#include <DallasTemperature.h>

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#include <PubSubClient.h>

#include <stdio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Code for ESP8266WiFiMulti.h

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef WIFICLIENTMULTI\_H\_

#define WIFICLIENTMULTI\_H\_

#include "ESP8266WiFi.h"

#include <vector>

#ifdef DEBUG\_ESP\_WIFI

#ifdef DEBUG\_ESP\_PORT

#define DEBUG\_WIFI\_MULTI(fmt, ...) DEBUG\_ESP\_PORT.printf\_P( (PGM\_P)PSTR(fmt), ##\_\_VA\_ARGS\_\_ )

#endif

#endif

#ifndef DEBUG\_WIFI\_MULTI

#define DEBUG\_WIFI\_MULTI(...) do { (void)0; } while (0)

#endif

struct WifiAPEntry

{

char \* ssid;

char \* passphrase;

};

typedef std::vector<WifiAPEntry> WifiAPlist;

class ESP8266WiFiMulti

{

public:

ESP8266WiFiMulti();

~ESP8266WiFiMulti();

bool addAP(const char\* ssid, const char \*passphrase = NULL);

bool existsAP(const char\* ssid, const char \*passphrase = NULL);

wl\_status\_t run(void);

void cleanAPlist(void);

private:

WifiAPlist APlist;

bool APlistAdd(const char\* ssid, const char \*passphrase = NULL);

bool APlistExists(const char\* ssid, const char \*passphrase = NULL);

void APlistClean(void);

};

#endif /\* WIFICLIENTMULTI\_H\_ \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Define Constants and Instances

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define TOKEN "\*\*\*\*\*\*\*\*\*\*\*\*" // Your Ubidots TOKEN

#define WIFINAME "\*\*\*\*\*" // Your SSID

#define WIFIPASS "\*\*\*\*\*\*\*\*\*i" // Your Wifi Pass

#define MQTTCLIENTNAME "\*\*\*\*\*\*" // Your MQTT Client Name, it must be unique so we recommend to choose a random ASCCI name

#define VARIABLE\_LABEL "position" // Assing the variable label

#define DEVICE\_LABEL "BELT" // Assig the device label

#define Pin D3

#define pulsePin A0

int temp;

static const int RXPin = 6, TXPin = 7;

static const uint32\_t GPSBaud = 9600;

// The TinyGPS++ object

TinyGPSPlus gps;

// The serial connection to the GPS device

SoftwareSerial ss(RXPin, TXPin);

OneWire ourWire(Pin);

DallasTemperature sensors(&ourWire);

Ubidots client(TOKEN,MQTTCLIENTNAME);

// Global variables and defines

int16\_t mpu6050Ax, mpu6050Ay, mpu6050Az;

int16\_t mpu6050Gx, mpu6050Gy, mpu6050Gz;

// object initialization

MPU6050 mpu6050;

// define vars for testing menu

const int timeout = 10000; //define timeout of 10 sec

char menuOption = 0;

long time0;

int rate[10];

unsigned long sampleCounter = 0;

unsigned long lastBeatTime = 0;

unsigned long lastTime = 0, N;

int BPM = 0;

int IBI = 0;

int P = 512;

int T = 512;

int thresh = 512;

int amp = 100;

int Signal;

boolean Pulse = false;

boolean firstBeat = true;

boolean secondBeat = true;

boolean QS = false;

char mqttBroker[] = "industrial.api.ubidots.com";

char payload[700];

char topic[150];

// Space to store values to send

char str\_val[6];

char str\_lat[6];

char str\_lng[6];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Auxiliar Functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void callback(char\* topic, byte\* payload, unsigned int length)

{

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("] ");

for (int i = 0; i < length; i++)

{

Serial.print((char)payload[i]);

}

Serial.println();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Main Functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup()

{

Serial.begin(9600);

client.wifiConnection(WIFINAME, WIFIPASS);

sensors.begin();

client.begin(callback);

Wire.begin();

mpu6050.initialize();

void loop()

{

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

if (!client.connected())

{

client.reconnect();

}

sensors.requestTemperatures();

//Prepare the sensor for reading

temp = sensors.getTempCByIndex(0);

Serial.print(sensors.getTempCByIndex(0));

mpu6050.getMotion6(&mpu6050Ax, &mpu6050Ay, &mpu6050Az, &mpu6050Gx, &mpu6050Gy, &mpu6050Gz);

//read accelerometer and gyroscope raw data in three axes

double mpu6050Temp = ((double)mpu6050.getTemperature() + 12412.0) / 340.0;

Serial.print("a/g-\t");

Serial.print(mpu6050Ax); Serial.print("\t");

Serial.print(mpu6050Ay); Serial.print("\t");

Serial.print(mpu6050Az); Serial.print("\t");

Serial.print(mpu6050Gx); Serial.print("\t");

Serial.print(mpu6050Gy); Serial.print("\t");

Serial.print(mpu6050Gz); Serial.print("\t");

Serial.print(F("Temp- "));

Serial.println(mpu6050Temp);

if (QS == true)

{

Serial.println("BPM: "+ String(BPM));

QS = false;

} else if (millis() >= (lastTime + 2))

{

readPulse();

lastTime = millis();

// client.add("pulse",BPM);

// delay(1000);

}

//Read and print

client.add("temperature", temp);

client.add("motion",mpu6050Az);

client.add("pulse",BPM);

//Insert your variable Labels and the value to be sent

delay(1000);

client.ubidotsPublish("IoTBelt");

client.loop();

float belt\_position = random(0, 9);

float lat =gps.location.lat();

float lng= gps.location.lng();

/\* 4 is mininum width, 2 is precision; float value is copied onto str\_temp\*/

dtostrf(belt\_position, 4, 2, str\_val);

dtostrf(lat, 4, 2, str\_lat);

dtostrf(lng, 4, 2, str\_lng);

sprintf(topic, "%s", ""); // Cleans the topic content

sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE\_LABEL);

sprintf(payload, "%s", ""); // Cleans the payload content

sprintf(payload, "{\"%s\":", VARIABLE\_LABEL); // Adds the variable label

sprintf(payload, "%s {\"value\": %s", payload, str\_val); // Adds the value

sprintf(payload, "%s, \"context\":{\"lat\": %s, \"lng\": %s}", payload, str\_lat, str\_lng); // Adds coordinates

sprintf(payload, "%s } }", payload); // Closes the dictionary brackets

client.ubidotsPublish(payload);

client.loop();

delay(1000);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Functions call for Pulse sensor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void readPulse()

{

Signal = analogRead(pulsePin);

sampleCounter += 2;

int N = sampleCounter - lastBeatTime;

detectSetHighLow();

if (N > 250)

{

if ( (Signal > thresh) && (Pulse == false) && (N > (IBI / 5) \* 3) )

pulseDetected();

}

if (Signal < thresh && Pulse == true)

{

Pulse = false;

amp = P - T;

T = thresh;

}

if (N > 2500)

{

thresh = 512;

P = 512;

T = 512;

lastBeatTime = sampleCounter;

firstBeat = true;

secondBeat = true;

}

}

void detectSetHighLow()

{

if (Signal < thresh && N > (IBI / 5) \* 3)

{

if (Signal < T)

{

T = Signal;

}

}

if (Signal > thresh && Signal > P)

{

P = Signal;

}

}

void pulseDetected()

{

Pulse = true;

IBI = sampleCounter - lastBeatTime;

lastBeatTime = sampleCounter;

if (firstBeat)

{

firstBeat = false;

return;

}

if (secondBeat)

{

secondBeat = false;

for (int i = 0; i <= 9; i++)

{

rate[i] = IBI;

}

}

word runningTotal = 0;

for (int i = 0; i <= 8; i++)

{

rate[i] = rate[i + 1];

runningTotal += rate[i];

}

rate[9] = IBI;

runningTotal += rate[9];

runningTotal /= 10;

BPM = 60000 / runningTotal;

QS = true;

}

**Drawback-:**

NodeMCU doesn't work when it is not in range of wifi.

**Conclusion-:**

In this Internship we were able to learn about IoT technology and how to interface different sensors with NodeMCU and display the output on ubidots.