PROJECT_WSN_2016_Group_21

CSD 337

Divya Lohani

Project Report

Environmental monitoring application : Magnetic Compass and Telegrambot (MCAT)

April 18, 2016

Magnetic Compass and Telegrambot (MCAT)

Domain:

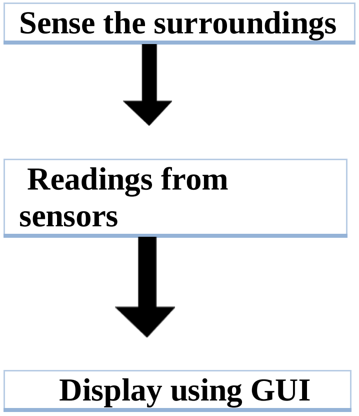
This project falls in the category of Environmental Monitoring as :

- one can find temperature, humidity, pressure and altitude values with the help of a telegram bot.
- Also, one can have a sense of direction with the help of HMC5883L sensor incorporated on a breadboard alongside LEDs that glow in accordance with the direction of the sensor's rotation.

Objective:

The aim is to obtain pressure, temperature and altitude values from one's surroundings in an efficient manner. The sense of direction in also incorporated to make the application smarter.

Flowchart:



Real World Applications:

- It is a smart application that can be used by ships to keep a track of direction while simultaneously allowing them to obtain pressure, temperature and humidity values of their surroundings.
- Can also be used by airplanes for the same purpose.

Hardware Used:

- Arduino UNO
- BMP 180
- DHT11
- HMC5883
- LEDs (4)
- Breadboard
- Jumper wires

Snapshots:

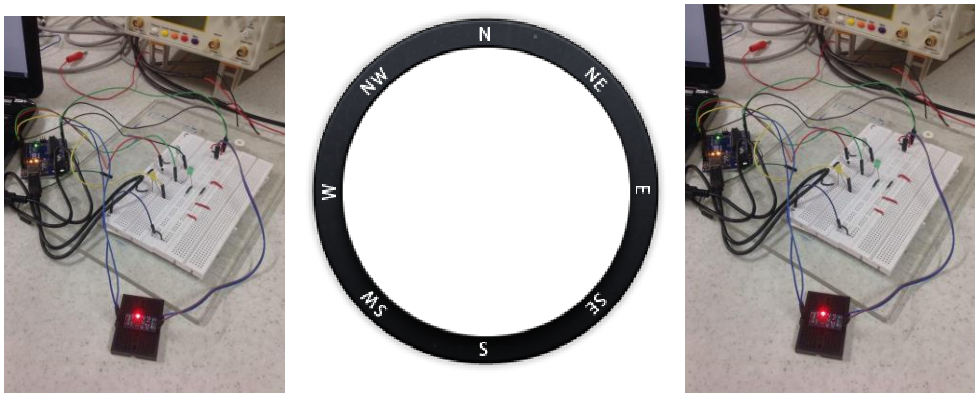


Fig 1. Images of HMC5883 connected to an Arduino UNO

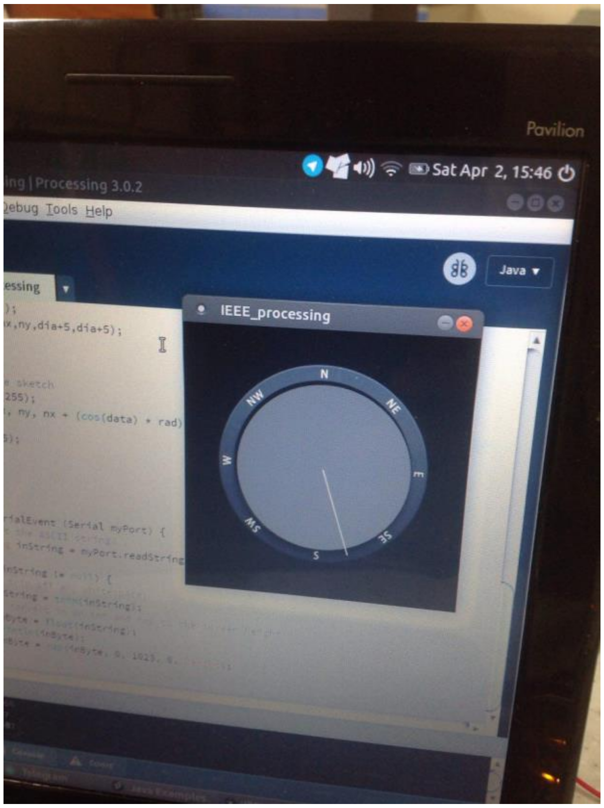
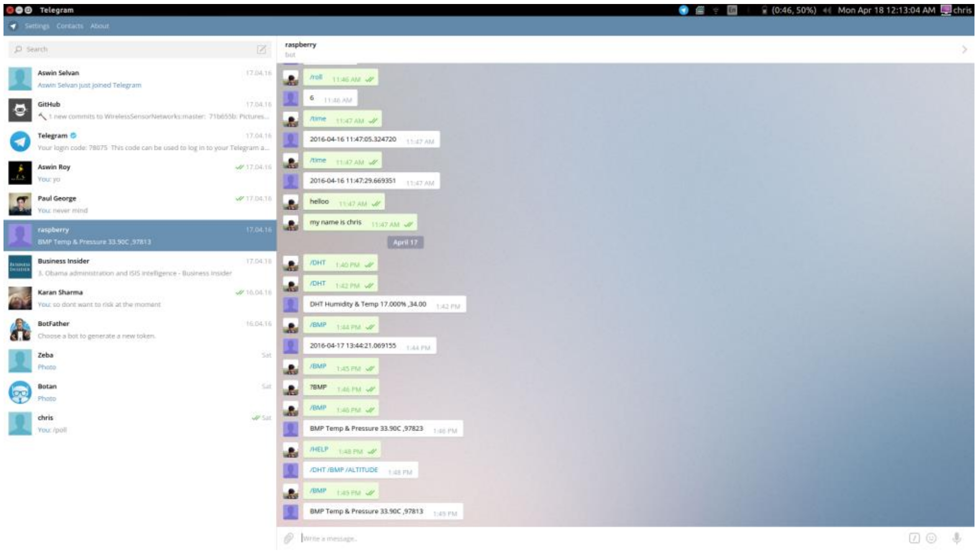


Fig 2. Processing GUI



Aswin Selvan 17.04.16
Aswin Selvan just joined Telegram

GitHub 17.04.16
1 new commits to WirelessSensorNetworks:master: 71b655b: Pictures...

Telegram 17.04.16
Your login code: 78075 This code can be used to log in to your Telegram a...

Aswin Roy 17.04.16
You: yo

Paul George 17.04.16
You: never mind

raspberry 17.04.16
BMP Temp & Pressure 33.90C ,97813

Business Insider 17.04.16
3. Obama administration and ISIS intelligence - Business Insider

Karan Sharma 16.04.16
You: so dont want to risk at the moment

BotFather 16.04.16
Choose a bot to generate a new token.

Zeba Sat
Photo

Botan Sat
Photo

chris Sat
You: /poll

/roll 11:46 AM

6 11:46 AM

/time 11:47 AM

2016-04-16 11:47:05.324720 11:47 AM

/time 11:47 AM

2016-04-16 11:47:29.669351 11:47 AM

helloo 11:47 AM

my name is chris 11:47 AM

April 17

/DHT 1:40 PM

/DHT 1:42 PM

DHT Humidity & Temp 17.000% ,34.00 1:42 PM

/BMP 1:44 PM

2016-04-17 13:44:21.069155 1:44 PM

/BMP 1:45 PM

?BMP 1:46 PM

/BMP 1:46 PM

BMP Temp & Pressure 33.90C ,97823 1:46 PM

/HELP 1:48 PM

/DHT /BMP /ALTITUDE 1:48 PM

/BMP 1:49 PM

BMP Temp & Pressure 33.90C ,97813 1:49 PM

Fig 3. Telegrambot responding to the commands to give sensors' readings

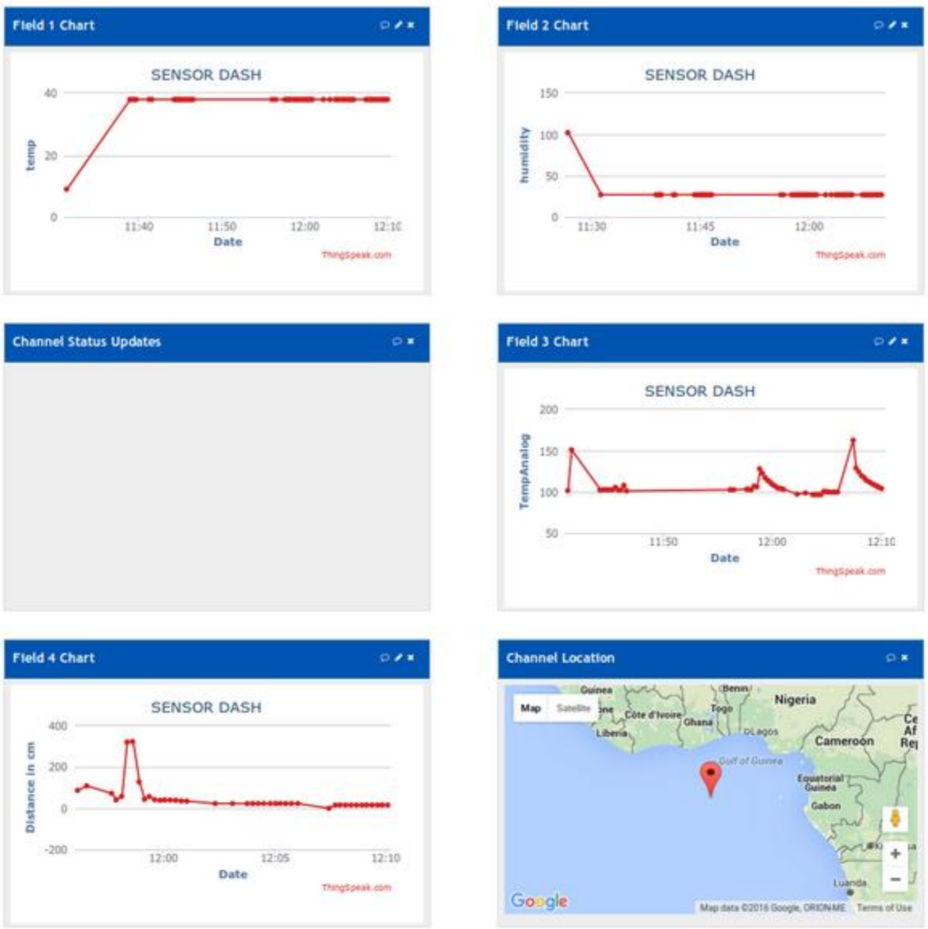


Fig 4. Graphs drawn using 'ThingsSpeak.com'

Main Code:

```
// Authors: Charmi Badlani, Chris Sunny, Arvind Satyamurthy
#include <Wire.h>
#include "HMC5883L.h"          //library

int LED1= 5;
int LED2= 10;
int LED3= 9;
int LED4= 3;

HMC5883L compass;

void setup()
```

```
{
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(LED3,OUTPUT);
  pinMode(LED4,OUTPUT);

  Serial.begin(9600);
  Serial.println("Initialize HMC5883L");
  while (!compass.begin())
  {
    Serial.println("Could not find a valid HMC5883L sensor, check wiring!");
    delay(500);
  }

  // Set measurement range
  compass.setRange(HMC5883L_RANGE_1_3GA);

  // Set measurement mode
  compass.setMeasurementMode(HMC5883L_CONTINOUS);

  // Set data rate
  compass.setDataRate(HMC5883L_DATARATE_30HZ);

  // Set number of samples averaged
  compass.setSamples(HMC5883L_SAMPLES_8);

  // Set calibration offset. See HMC5883L_calibration.ino
  compass.setOffset(0, 0);
}

void loop()
{
  Vector norm = compass.readNormalize();

  // Calculate heading
  float heading = atan2(norm.YAxis, norm.XAxis);

  // Set declination angle on your location and fix heading
  // You can find your declination on: http://magnetic-declination.com/
  // (+) Positive or (-) for negative
  // For Bytom / Poland declination angle is 4'26E (positive)
  // Formula: (deg + (min / 60.0)) / (180 / M_PI);
  float declinationAngle = (4.0 + (26.0 / 60.0)) / (180 / M_PI);
  heading += declinationAngle;

  // Correct for heading < 0deg and heading > 360deg
  if (heading < 0)
  {
    heading += 2 * PI;
  }

  if (heading > 2 * PI)
  {
    heading -= 2 * PI;
  }

  // Convert to degrees
  float headingDegrees = heading * 180/M_PI;
`  Serial.println(headingDegrees);
   dir_glow(headingDegrees);
}



void dir_glow(int value){                                    // function to glow led based on the sensor
readings
  if( (value>=350 && value <=359.9) || (value >=0 && value<=10) ){
    digitalWrite(LED1,HIGH);
  }
  else{
     digitalWrite(LED1,LOW);
  }

  if(value >=80 && value <=100){
    digitalWrite(LED2,HIGH);
  }
  else {
    digitalWrite(LED2,LOW);
  }

  if(value >=170 && value <=190){
   digitalWrite(LED3,HIGH);
  }
  else{
    digitalWrite(LED3,LOW);
  }

  if(value >=260 && value <=280){
    digitalWrite(LED4,HIGH);
  }
  else{
    digitalWrite(LED4,LOW);
  }
}

// Processing GUI

import processing.serial.*;

Serial myPort;                 // Create object from Serial class
float inByte =0;
float data;                    // Data received from the serial port in degrees
float rad;
float needle;
float dia;
float nx,ny;
PImage imgCompassRing;
float offset ;

void setup(){                  //setup function
  size(300,300);

  String portName = Serial.list()[0];
  myPort = new Serial(this, portName, 9600);
  myPort.bufferUntil('\n');
  stroke(255);

  //rad = min(width, height) / 2;
  rad = 100;
  needle= rad * 0.60;
  dia = rad * 1.8;

  nx= (width/2);
  ny=(height/2);

}

void draw(){                    //draw function
 background(0);
 imgCompassRing = loadImage("compassRing.png");
 image(imgCompassRing, 18.5, 18);
 if ( myPort.available() > 0) {  // If data is available,
    data= myPort.read();         // read it and store it in val
    println(data);
  }
 fill(80);                       // compass background
 noStroke();
 ellipse(nx,ny,dia+5,dia+5);



                              //needle sketch
  stroke(255);
  line(nx, ny, nx + (cos(data) * rad), ny + (sin(data) * rad));

  delay(5);
}


void serialEvent (Serial myPort) {
  // get the ASCII string:
  String inString = myPort.readStringUntil('\n');

  if (inString != null) {
    // trim off any whitespace:
    inString = trim(inString);
    // convert to an int and map to the screen height:
    inByte = float(inString);
    //println(inByte);
    //inByte = map(inByte, 0, 360, 0, height);
  }}
```