Estación meteorológica
Proyecto final
Instrumentación

Carlos López Roa
CFATA-UNAM

13 de junio de 2012

1. Introducción

En esta materia se requiere presentar un proyecto final que demuestre los conocimientos adquiridos en el curso y por sobre todas las cosas FUNCIONE.

En este trabajo se presenta la realización, desde concepto hasta implementación, de una estación meteorológica sobre un globo aerostático utilizando los siguiente componentes: Tarjeta de desarrollo Arduino Uno, Sensores de presión, temperatura, aceleración operados con el protocolo I^2C, Tarjetas inalámbricas Xbee para el envío de datos de forma serial, software de visualización Processing.

Palabras claves: Arduino, Temperatura, Presión, Acelerómetro, XBee, Processing

2. Objetivo

Instrumentar una estación meteorológica sobre un globo aerostático, que envíe mediciones del tiempo inalámbricamente a una computadora en tierra.

3. Desarrollo

3.1. Descripción general

Entonces, queremos montar una estación meteorológica sobre un globo aerostático y comunicarla a tierra inalámbricamente. El globo aerostático se logró utilizando un globo relleno de gas Helio, el gas He es el más ligero después del Hidrógeno pero es mucho menos reactivo. A este globo se le anexa una canasta donde estarán los electrónicos, así como unos ventiladores comerciales a Radio control para su manipulación en el aire. El sistema de muestreo consiste en dos sensores: Uno de presión y temperatura (BMP085 breakout) y un acelerómetro (MMA8452) que se comunican por el protocolo I^2C con una tarjeta de desarrollo Arduino Uno. La tarjeta Arduino establece comunicación RS232 con módulos de comunicación inalámbrica $Xbee$. Los datos enviados de un $Xbee$ a otro se recogen en tierra con otra tarjeta Arduino, que establece comunicación serial con una computadora. La computadora recoge los datos y los despliega en Processing.

3.2. Los sensores

Se adquirieron dos sensores, hojas de datos anexas, con las siguientes características generales: [4]

Sensor	Modelo	Marca	Resolución	Voltaje de entrada
Temperatura y Presión	BMP085	Bosch	300-1100hPa	1.8-3.6V
Aceleración	MMA8452Q	Freescale	± 2,4,8 g	1.95-3.6V

Cuadro 1: Tabla general de los sensores usados [1, 2]

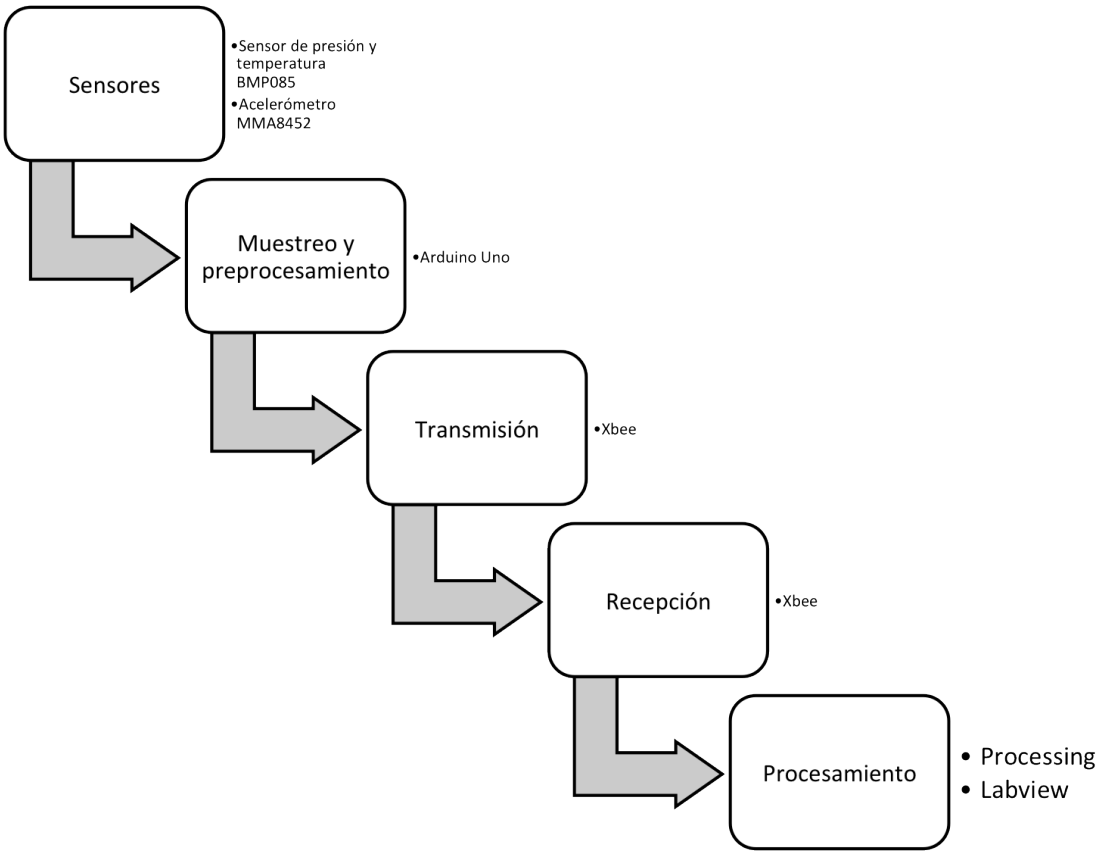

Figura 1: Esquema general del proyecto

Ambos sensores manejan el protocolo I^2C por medio del cual ambos poseen una dirección hexadecimal y por lo tanto ambos pueden comunicarse al microcontrolador por la misma vía. Dentro del protocolo I^2C existe el siguiente esquema general de comunicación [3]

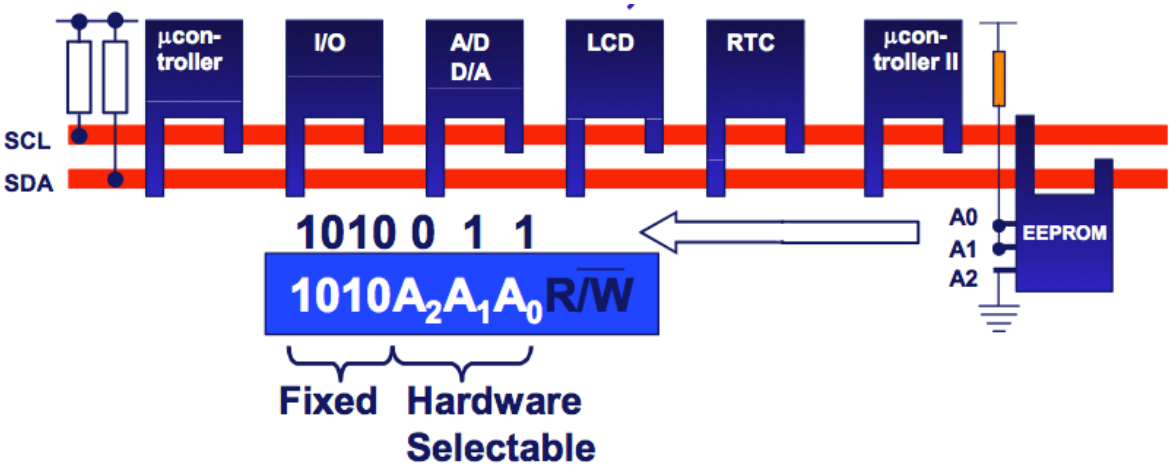

Figura 2: Esquema general de direccionamiento en el protocolo I^2C [3]

En este protocolo hace falta establecer una dirección única para cada dispositivo en la red.

Así como enviar códigos de inicio de comunicación, esperar a una respuesta y enviar una dirección de lectura o escritura en el dispositivo, al final enviar un código de fin de la comunicación.

Ambas características se lograron implementando los códigos de ejemplo del fabricante, así como las bibliotecas necesarias para su implementación en Arduino. Ambas anexas [4]

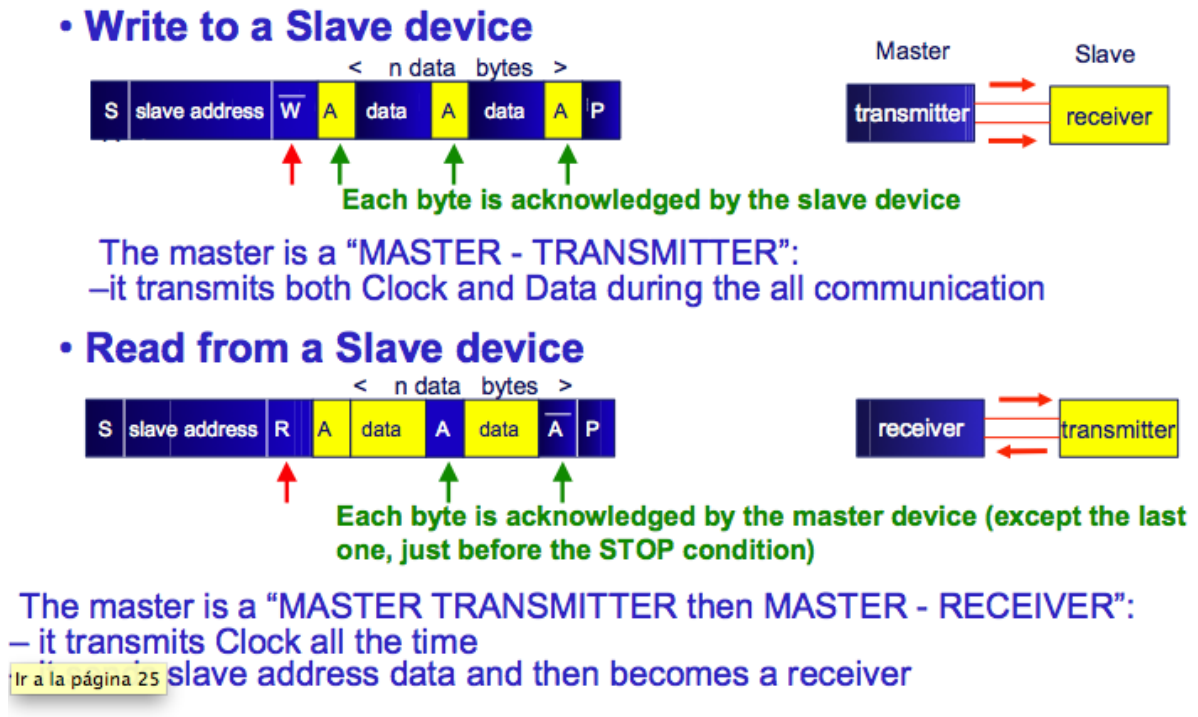

Figura 3: Operaciones de lectura y escritura en el protocolo I^2C [3]

3.3. Muestreo y preprocesamiento

Para lograr el muestreo utilizando el Arduino se desarrolló una tarjeta Arduino Shield utilizando las bibliotecas de Eagle anexas y el siguiente esquema: [4]

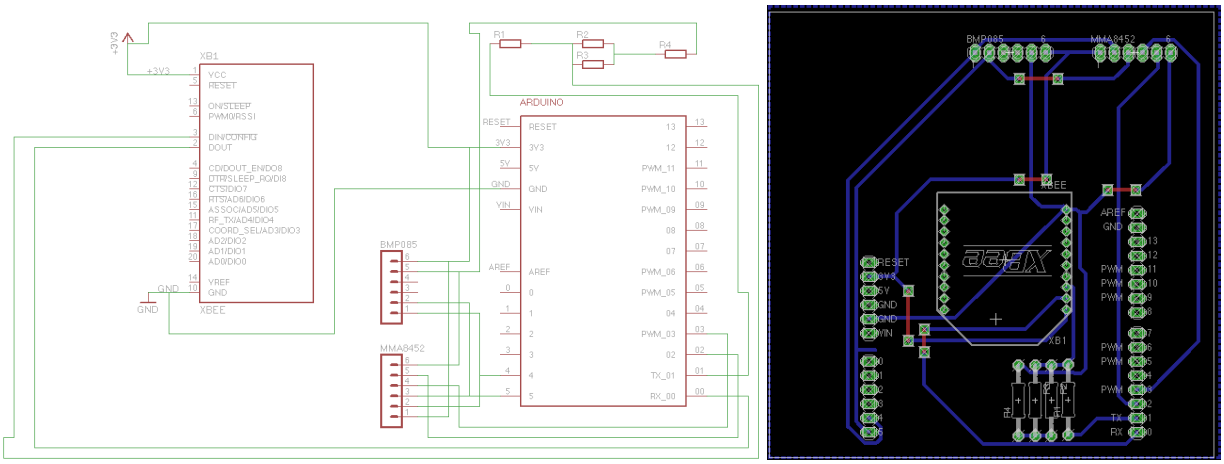

Figura 4: Esquemático y PCB del Arduino Shield construido.

Este PCB se monta sobre la placa de desarrollo y envía datos siguiente la rutina de muestreo.

La rutina de muestreo y preprocesamiento de las señales se describe en el anexo 1. En esta rutina se estableció el envío de un String separado por tabulaciones donde los primeros tres valores corresponden a los valores de aceleración en gravedades en el eje x,y,z respectivamente, el cuarto valor corresponde al valor de la temperatura multiplicado por 10 y el último al valor de presión en Pascales.

El Arduino se alimenta abordo a 6V con 4 baterías AA.

3.4. Transmisión

El módulo Xbee recibe los datos por los pines *TX,RX* de la placa y los envía inalámbricamente a su esclavo.

La comunicación serial del módulo *Xbee* se hace a 3.3V para economizar energía, por lo que se implementó un divisor de voltaje.

3.5. Recepción

Otro módulo *Xbee* en tierra y conectado similarmente a otra placa Arduino recibe los datos y los trasmite por vía serial con el código mostrado en el anexo 3. Este programa implementa un escuchador de eventos en el puerto serial, espera a su recepción y lo despliega en el puerto serial como un String.

3.6. Procesamiento

3.6.1. Processing

Este programa, desarrollado por los creadores de Arduino, escucha el puerto serial, captura una línea a la vez y concatena los valores en arreglos. Posteriormente mapea la lectura de un rango especificado al rango de la pantalla para graficar los 5 valores al mismo tiempo y en el mismo plano. Este programa se exhibe en el anexo 2

4. Conclusiones

- Se logró instrumentar un globo aerostático con una estación de monitoreo que envía inalámbricamente los datos adquiridos a una estación en tierra que posteriormente procesa lo adquirido.
- La interfaz de comunicación I^2C resulta ideal para implementar distintos sensores pues permite la comunicación por sólo dos canales.
- Un sistema embebido con las características del Arduino resulta muy conveniente para esta aplicación, su programación es una variante de C y C++. Aunque pudo utilizarse cualquier otro microcontrolador.
- La interfaz Processing es una variante de Java y resultó sencilla de implementar.

5. Anexos

5.1. Programa abordo del Arduino

```
/*  BMP08
 SDA             A4.
 SCL             A5,
 XCLR            --
 EOC             D2
 GND             GND
 VCC             VCC

 MMA8452 Breakout -------- Arduino
 3.3V --------------------- 3.3V
 SDA ---------------------- A4
 SCL ---------------------- A5
 INT2 --------------------- D3
 INT1 --------------------- D2
 GND ---------------------- GND
 */

#include "i2c.h"
#include <Wire.h>

```

```
#define SA0 1
#if SA0
#define MMA8452_ADDRESS 0x1D
#else
#define MMA8452_ADDRESS 0x1C
#endif

#define I2C_ADDRESS 0x77
#define BMP085_ADDRESS 0x77

const unsigned char OSS = 0;
int ac1; int ac2; int ac3;
unsigned int ac4; unsigned  int ac5; unsigned int ac6;
int b1;  int b2;  int mb; int mc; int md;
long b5; short temperature; long pressure;
const byte scale = 2; const byte dataRate = 0;
int int1Pin = 2; int int2Pin = 3;
byte data[6]; int accelCount[3];  float accel[3]; float a;
float an = 0; int t = 100;

void setup(){
    Serial.begin(9600);
    mma8452setup();
    bmp08setup();

}// Fin de Setup

void loop(){
    mma8452loop();
    bmp08loop();
    Serial.println();
    delay(t);

}//Fin de loop

void bmp08setup(){
    bmp085Calibration();
    Serial.println("BMP085 Encontrado");

}// Fin de bmp08setup

void mma8452setup(){
    byte c;

    pinMode(int1Pin, INPUT);
    digitalWrite(int1Pin, LOW);
    pinMode(int2Pin, INPUT);
    digitalWrite(int2Pin, LOW);

    c = readRegister(0x0D);
    if (c == 0x2A) {
        initMMA8452(scale, dataRate);
        Serial.println("MMA8452Q encontrado");
    }
    else{
        Serial.print("No se puede conectar a MMA8452Q: 0x");
        Serial.println(c, HEX);
        while (1)
```

```
            ;
    }
}//Fin de mma8452setup

void bmp08loop(){
    temperature = bmp085GetTemperature(bmp085ReadUT());
    pressure = bmp085GetPressure(bmp085ReadUP());
    Serial.print(temperature, DEC);
    Serial.print("\t");

    Serial.print(pressure, DEC);

}//Fin de bmp08loop

void mma8452loop(){
    static byte source;

    if (digitalRead(int1Pin))  {
        readRegisters(0x01, 6, &data[0]);

        for (int i=0; i<6; i+=2) {
            accelCount[i/2] = ((data[i] << 8) | data[i+1]) >> 4;
            if (data[i] > 0x7F) {
                accelCount[i/2] = ~accelCount[i/2] + 1;
                accelCount[i/2] *= -1;
            }
            accel[i/2] = (float) accelCount[i/2]/((1<<12)/(2*scale));
        }
        a=0;
        for (int i=0; i<3; i++) {
            Serial.print(accel[i],10);
            Serial.print('\t');
        }
    }//Fin de interrupcion 1
}//Fin de mma8452loop
```

5.2. Programa fuera de borda en Processing

```
import processing.serial.*;
Serial arduino;

String stringAccX;
String stringAccY;
String stringAccZ;
String stringTemp;
String stringPres;

int[] accX = new int[600];
float inAccX;
int[] accY = new int[600];
float inAccY;
int[] accZ = new int[600];
float inAccZ;
int[] Temp = new int[600];
float inTemp;
int[] Pres = new int[600];
float inPres;
```

```
20
21
22 void setup()
23 {
24   size(600,400);
25   println(arduino.list());
26
27   arduino.bufferUntil('\n');
28
29   for(int i=0;i<600;i++)
30   {
31    accX[i] = height/2;
32    accY[i] = height/2;
33    accZ[i] = height/2;
34    Temp[i] = height/2;
35    Pres[i] = height/2;
36   }
37 }
38
39 void draw()
40 {
41   background(255);
42   for(int i = 0 ;i<=width/10;i++)
43   {
44     stroke(200);
45     line((-frameCount%10)+i*10,0,(-frameCount%10)+i*10,height);
46     line(0,i*10,width,i*10);
47   }
48   convert();
49   drawAxisX();
50   drawAxisY();
51   drawAxisZ();
52   drawAxisT();
53   drawAxisP();
54 }
55
56
57 void serialEvent (Serial arduino)
58 {
59  stringAccX = arduino.readStringUntil('\t');
60  stringAccY = arduino.readStringUntil('\t');
61  stringAccZ = arduino.readStringUntil('\t');
62  stringTemp = arduino.readStringUntil('\t');
63  stringPres = arduino.readStringUntil('\n');
64
65     printAxis();
66   printConvert();
67
68 }
69
70 int maxAngle = 90;
71 int maxG = 2;
72 int maxTemp = 400;
73 int minTemp = 200;
74 int maxPres = 110000;
75 int minPres = 300;
76
77 void convert()
```

```
{
 if (stringAccX != null)
 {
   stringAccX = trim(stringAccX);
   inAccX = float(stringAccX);
   inAccX = map(inAccX, maxG, -maxG, 0, height);
   accX[accX.length-1] = int(inAccX);
 }
 if (stringAccY != null)
 {
   stringAccY = trim(stringAccY);
   inAccY = float(stringAccY);
   inAccY = map(inAccY, maxG, -maxG, 0, height);
   accY[accY.length-1] = int(inAccY);
 }
 if (stringAccZ != null)
 {
   stringAccZ = trim(stringAccZ);
   inAccZ = float(stringAccZ);
   inAccZ = map(inAccZ, maxG, -maxG, 0, height);
   accZ[accZ.length-1] = int(inAccZ);
 }
  if (stringTemp != null)
 {
   stringTemp = trim(stringTemp);
   inTemp = float(stringTemp);
   inTemp = map(inTemp, maxTemp, minTemp , 0, height);
   Temp[Temp.length-1] = int(inTemp);
 }
   if (stringPres != null)
 {
   stringPres = trim(stringPres);
   inPres = float(stringPres);
   inPres = map(inPres, maxPres, minPres, 0, height);
   Pres[Pres.length-1] = int(inPres);
 }
}//Fin de Convert

void drawAxisX()
{
  noFill();
  stroke(129,129,230);
  beginShape();
  for(int i = 0; i<accX.length;i++){
    vertex(i,accX[i]);
  }
  endShape();
  for(int i = 1; i<accX.length;i++){
   accX[i-1] = accX[i];
   }
}

void drawAxisY()
{
  noFill();
  stroke(127,34,255);
  beginShape();
  for(int i = 0; i<accY.length;i++){
```

```
136     vertex(i,accY[i]);
137   }
138   endShape();
139   for(int i = 1; i<accY.length;i++){
140    accY[i-1] = accY[i];
141    }
142 
143 }//Fin de drawAxisY
144 
145 void drawAxisZ()
146 {
147   noFill();
148   stroke(255,0,0);
149   beginShape();
150   for(int i = 0; i<accZ.length;i++){
151     vertex(i,accZ[i]);
152   }
153   endShape();
154   for(int i = 1; i<accZ.length;i++){
155    accZ[i-1] = accZ[i];
156    }
157 
158 }//Fin de drawAxisZ
159 
160 void drawAxisT()
161 {
162   noFill();
163   stroke(0,255,0);
164   beginShape();
165   for(int i = 0; i<Temp.length;i++){
166     vertex(i,Temp[i]);
167   }
168   endShape();
169   for(int i = 1; i<Temp.length;i++){
170    Temp[i-1] = Temp[i];
171    }
172 
173 }//Fin de drawAxisT
174 
175 void drawAxisP()
176 {
177   noFill();
178   stroke(0,255,255);
179   beginShape();
180   for(int i = 0; i<Pres.length;i++){
181     vertex(i,Pres[i]);
182   }
183   endShape();
184   for(int i = 1; i<Pres.length;i++){
185    Pres[i-1] = Pres[i];
186    }
187 
188 }//Fin de drawAxisP
189 
190 void printAxis()
191 {
192  print(stringAccX);
193  print(stringAccY);
```

```
 print(stringAccZ);
 print(stringTemp);
 print(stringPres);
}

void printConvert()
{
  print(accX[accX.length-1]);
  print(",");
  print(accY[accY.length-1]);
  print(",");
  print(accZ[accZ.length-1]);
  print(",");
  print(Temp[Temp.length-1]);
  print(",");
  print(Pres[Pres.length-1]);
  println();
}
```

5.3. Programa fuera de borda del Arduino

```
/*
  Serial Event example

 When new serial data arrives, this sketch adds it to a String.
 When a newline is received, the loop prints the string and
 clears it.

 A good test for this is to try it with a GPS receiver
 that sends out NMEA 0183 sentences.

 Created 9 May 2011
 by Tom Igoe

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/SerialEvent

 */

String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // reserve 200 bytes for the inputString:
  inputString.reserve(200);
}

void loop() {
  // print the string when a newline arrives:
  if (stringComplete) {
    Serial.println(inputString);
    // clear the string:
    inputString = "";
    stringComplete = false;
```

```
37    }
38  }
39
40  /*
41    SerialEvent occurs whenever a new data comes in the
42   hardware serial RX.  This routine is run between each
43   time loop() runs, so using delay inside loop can delay
44   response.  Multiple bytes of data may be available.
45   */
46  void serialEvent() {
47    while (Serial.available()) {
48      // get the new byte:
49      char inChar = (char)Serial.read();
50      // add it to the inputString:
51      inputString += inChar;
52      // if the incoming character is a newline, set a flag
53      // so the main loop can do something about it:
54      if (inChar == '\n') {
55        stringComplete = true;
56      }
57    }
58  }
```

5.4. Cronograma y *log* de actividades

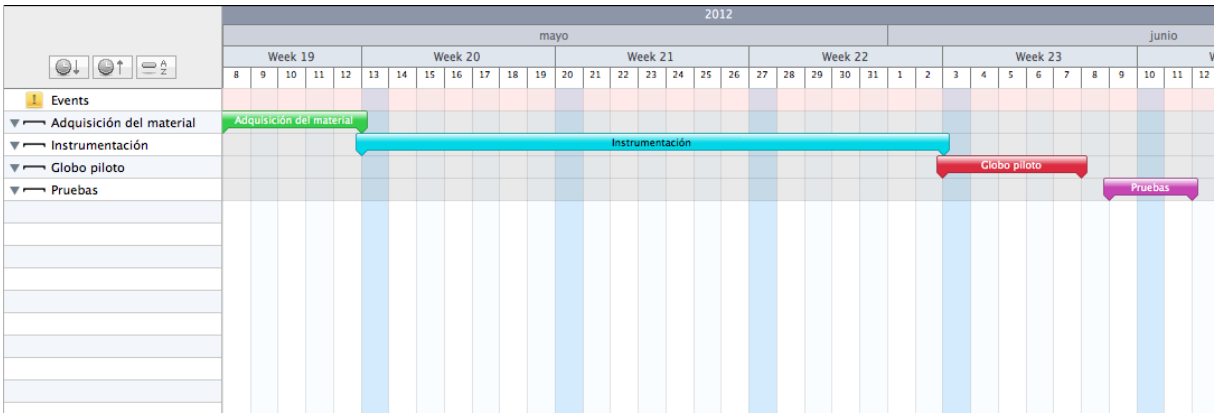

Figura 5: Planeación original del proyecto planteada el 8 de mayo 2012

5.5. Registro de actividades

- 16/05/12 Se platicó con el profesor Rafael sobre el esquema general del proyecto. Surge la idea de "Globo aerostático como estación meteorológica"
- 21/05/12 Se adquirió el material

  1 Termometro Hidrometro Higrometro (402725091). Financiación: 1 mensualidad/es de $ 199.00

  4 Acelerómetro analógico de 3 e $ 260.00 6 Acelerómetro digital de 3 eje $ 170.00 1 Barometric Pressure Sensor - $ 300.00

  Subtotal: $ 2,360.00 Costo de envio (México (con segur: $ 224.40 Total (MXN): $ 2,584.40

  Con los distintos proveedores.

- 23/05/12 Se recibió el material enviado por: Mensajería: Estafeta Número de guía: 0039803445
- 28/05/12 Se adquirió el material 1 Tarjeta De Desarrollo Arduino Uno (403038270). Financiación: 1 mensualidad/es de $ 459.99
- 29/05/12 Se logró establecer comunicación con el sensor de aceleración MMA8452Q utilizando el protocolo I2C y la tarjeta de desarrollo Arduino. Se investigan algoritmos para la estimación de la velocidad y el desplazamiento a partir de medidas en la aceleración.
- 30/05/12

  Se presentaron los avances con el doctor. Se estableció el esquema general de vuelo. Un globo aerostático sostenido por Helio, con motores controlados vía RC. La tarjeta Arduino muestrea los distintos sensores, establece comunicación RS232 con el emisor Xbee mismo que reporta al receptor una cadena de caracteres con las mediciones en tierra. El string es decodificado por software y despliega las medidas. Se espera un pasajero a bordo. Se pudo establecer comunicación con el sensor BMP085 utilizando el protocolo I2C
- 31/05/12 Se programó la rutina general de muestreo, donde ambos sensores I2C comparten vías de comunicación pero ambos envían sus datos a distintos tiempos.
- 7/06/12

  Se programó la rutina de obtención de datos y graficado en Processing. Se obtuvieron diagramas PCB para la creación de un shield para arduino personalizado. Se necesita trabajar en Eagle. Todo listo para la comunicación serial a Xbee.
- 11/06/12

  Se adquirió el material para la realización del Arduino Shield que soportará los materiales sobre un circuito impreso para mayor estabilidad. Se probó el correcto funcionamiento de transmisión y recepción del Xbee. Se adquirieron juguetes diversos para usarlos de soporte.
- 12/06/12

  Se realizó el diseño del Arduino Shield en Eagle, se imprimió y transfirió correctamente a circuito impreso, se perforó y se soldaron las componentes necesarias. Es funcional.
- 13/06/12

  Se realizó reporte de actividades, se ensamblaron las piezas finales así como el gas Helio en el globo.

Referencias

[1] BMP085, Digital pressure sensor Data sheet, Bosch, Bosch Sensortec, 1.3, BST-BMP085-DS000-06, 0 273 300 144

[2] 3-Axis, 12-bit/8-bit Digital Accelerometer, Data Sheet: Technical Data, Freescale Semiconductor, MMA8452Q, Rev 4.1, 08/2011

[3] AN10216-01 I2C MANUAL, APPLICATION NOTE, Jean-Marc Irazabal , Steve Blozis, Philips Semiconductors, 2003

[4] Todos los anexos de este archivo están disponibles en `https://dl.dropbox.com/u/2115508/Anexos.zip`