FITNESS BAND

TABLE OF CONTENTS

ABSTRACT

The fitness of human being has always been a subject of interest such as sports, health, computer, and virtual applications. This project is motivated by the combination of public awareness on daily activities that promotes a healthy lifestyle and the increasing number of smart phone users.

This project incorporates a system based upon the smart phone, Arduino which are used to recognize different heartbeats, temperature, track users foot step, distance and number of steps. An Inertial Measurement Unit (IMU) which consists of a gyroscope and accelerometer sensor, heart beat sensor, temperature sensor (LM35) and Arduino is attached to a microcontroller and the data is transmitted to an Android mobile device via Bluetooth and computer connection to allow real time tracking functions.

This band really helps in keeping track of various things which is helpful in field of fitness and is in great demand these days.

Introduction

Now a days everybody wants to be fit so we are designing a fitness band which will help people to get the details of their heart rate, number of steps walked, temperature of the body, speed and etc. we will use arduino UNO as the micro controller with heartbeat sensor, accelero-gyro sensor, temperature sensor. These sensors will calculate heartbeat, speed, angular velocity, and body temperature. The device will be connected to an android smartphone in which it will show the results. The working of the application is done on Android studio. And the arduino coding is done on the Arduino web editor then it is debugged and exported to the arduino. This device will help everybody to get a regular check-up on their health.

FitnessBand that is implemented by Arduino has only one feature, which is collecting data by using accelerometer and sending them to a mobile device. Then the mobile device calculates the calories and steps by using the data. The feature is simple.

The android app check steps using collected data provided from FitnessBand Arduino. The algorithm of the app is not that complicated. If you have much experience to this area, one can replace it with your own algorithm. The app saves the calorie data, so you can see the progress it in a monthly/daily/hourly graph form. FitnessBand Arduino cannot save the data itself since the shortage of its memory capacity. That is, it only works when it’s connected with a mobile device.

Literature Survey

Fitness trackers seem to have gone mainstream — one in 10 adults in the United States now owns a fitness band. The survey of 5,000 U.S. adults, released Jan. 6, shows that people across the country are buying fitness trackers, which monitor activities such as steps taken, calories burned and time slept. The report found that 36 percent of people who own a fitness tracker are between 35 and 54 years old, 41 percent have an average income of more than $100,000 and 54 percent are women.

In contrast, just 2 percent of people in the U.S. own a smart-watch, and smart-watch owners are younger and less wealthy than fitness tracker wearers, according to the report. Most (69 percent) of people who own a smartwatch are ages 18 to 34, 48 percent earn less than $45,000 yearly and 71 percent are male.

When fitness trackers first hit stores in early 2013, "fitness fanatics and athletes" bought them in droves, Henderek told Live Science.

Now, two years later, more companies and improved products have entered the fitness tracker market. With increased awareness and choices, people — even those who only exercise casually — are buying fitness trackers.

Implementation

Step 1: Mechanism

FitnessBand consists of an Arduino part and an Android app.

The Arduino has 7 main parts –

- Arduino UNO
- Accelerometer(MPU-6050)
- Bluetooth module(HC-05)
- Temperature sensor(LM35)
- Heart beat pulse sensor
- Male-to-male jumper wires
- USB cable

Arduino UNO

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the

reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

Accelero-gyro sensor

GY-521 MPU6050 Module 3 Axis analog gyro sensors+ 3 Axis Accelerometer Module, MPU-6050 3 Axis Gyroscope Module for Arduino.

Features:

1. MPU-6050 modules (three-axis gyroscope + three-axis accelerometer)
2. Use the chip: MPU-6050
3. Power supply :3-5v (internal low dropout regulator)

Chip built-in 16bit AD converter, 16-bit.

Bluetooth

Most common Bluetooth modules that you can get are HC-05 main module and the one with interface base board. The latter one has a reset button, the status LED, and it supports both operation voltage(3,3v/5v), so this one is more convenient but the size is rather big, the LED, which is not quite necessary drains the battery.

Temperature Sensor

The **LM35** is an integrated circuit sensor that can be used to measure temperature with an electrical output proportional to the temperature (in ºC).

Use of LM35s To Measure Temperature

- can measure temperature more accurately than a using a thermistor.
- The sensor circuitry is sealed and not subject to oxidation, etc.

Heart beat sensor

The basic heartbeat sensor consists of a light emitting diode and a detector like a light detecting resistor or a photodiode.

- The heart beat pulses causes a variation in the flow of blood to different regions of the body. When a tissue is illuminated with the light source, i.e. light emitted by the led, it either reflects (a finger tissue) or transmits the light (earlobe). Some of the light is absorbed by the blood and the transmitted or the reflected light is received by the light detector.
- The amount of light absorbed depends on the blood volume in that tissue.
- The detector output is in form of electrical signal and is proportional to the heart beat rate.

Male-to-male jumper wires

It is used to connect the sensors with Arduino.

USB cables

It is used to connect Arduino with the computer.

Step- 2 Connections

Connecting Arduino-Bluetooth module

To manage Bluetooth module and test. Connect VCC, GND, TXD, RXD pin as the module in the instruction (VCC->3.3v, GND->GND, TX->D2, RX->D3).

Connecting Arduino-Accelerometer (MPU-6050)

An accelerometer module uses I2C interface. (VCC->3.3v, GND->GND, SDA->A4, SCL->A5)

Connecting Arduino-Temperature Sensor (LM-35)

For this, the GND is connected to D9, VCC-> D7 and the output pin is connected to A0.

Connecting Arduino-Heart beat Sensor

For this, the GND is connected to GND of Arduino, VCC->D6 and the output pin is connected to A1.

Connecting Arduino to Computer

To connect the setup to computer, USB cable is used.

Step 3: Arduino Source Code for Band

```
#include <Plotter.h>
```

```
#include <math.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include "Plotter.h"
/* Bluetooth */
SoftwareSerial BTSerial(2, 3); //Connect HC-06. Use your (TX, RX) settings
/* time */
#define SENDING_INTERVAL 1000
#define SENSOR_READ_INTERVAL 50
unsigned long prevSensoredTime = 0;
unsigned long curSensoredTime = 0;
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogInPin1 = A1;
const int analogOutPin1 = 10;
const int analogOutPin = 8; // Analog output pin that the LED is attached to
int sensorValue = 0;        // value read from the pot
int outputValue = 0;
int sensorValue1 = 0;        // value read from the pot
int outputValue1 = 0;
float temp;// value output to the PWM (analog out)
/* Data buffer */
#define ACCEL_BUFFER_COUNT 125
byte aAccelBuffer[ACCEL_BUFFER_COUNT];
int iAccelIndex = 2;
/* MPU-6050 sensor */
#define MPU6050_ACCEL_XOUT_H 0x3B // R
#define MPU6050_PWR_MGMT_1 0x6B // R/W
#define MPU6050_PWR_MGMT_2 0x6C // R/W
#define MPU6050_WHO_AM_I 0x75 // R
```

```
#define MPU6050_I2C_ADDRESS 0x68
typedef union accel_t_gyro_union {
  struct {
    uint8_t x_accel_h;
    uint8_t x_accel_l;
    uint8_t y_accel_h;
    uint8_t y_accel_l;
    uint8_t z_accel_h;
    uint8_t z_accel_l;
    uint8_t t_h;
    uint8_t t_l;
    uint8_t x_gyro_h;
    uint8_t x_gyro_l;
    uint8_t y_gyro_h;
    uint8_t y_gyro_l;
    uint8_t z_gyro_h;
    uint8_t z_gyro_l;
  } reg;
  struct {
    int x_accel;
    int y_accel;
    int z_accel;
    int temperature;
    int x_gyro;
    int y_gyro;
    int z_gyro;
  } value;
};
void setup() {
  int error;
```

```
uint8_t c;
digitalWrite(6,HIGH);
digitalWrite(7,HIGH);
digitalWrite(9,LOW);
Serial.begin(9600);
Wire.begin();
BTSerial.begin(9600);  // set the data rate for the BT port
// default at power-up:
// Gyro at 250 degrees second
// Acceleration at 2g
// Clock source at internal 8MHz
// The device is in sleep mode.
//
error = MPU6050_read (MPU6050_WHO_AM_I, &c, 1);
Serial.print(F("WHO_AM_I : "));
Serial.print(c,HEX);
Serial.print(F(", error = "));
Serial.println(error,DEC);
// According to the datasheet, the 'sleep' bit
// should read a '1'. But I read a '0'.
// That bit has to be cleared, since the sensor
// is in sleep mode at power-up. Even if the
// bit reads '0'.
error = MPU6050_read (MPU6050_PWR_MGMT_2, &c, 1);
Serial.print(F("PWR_MGMT_2 : "));
Serial.print(c,HEX);
Serial.print(F(", error = "));
Serial.println(error,DEC);
// Clear the 'sleep' bit to start the sensor.
MPU6050_write_reg (MPU6050_PWR_MGMT_1, 0);
```

```
  initBuffer();
}
void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);
  temp= ( analogRead(A0)-350) / 10;
  // print the results to the serial monitor:
  Serial.print("\t output(temp) = ");
  Serial.println(temp);
  Serial.print("TEMP = ");
  Serial.print(sensorValue);
  sensorValue1 = analogRead(analogInPin1);
  // map it to the range of the analog out:
  outputValue1 = map(sensorValue1, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin1, outputValue1);
  // print the results to the serial monitor:
  Serial.print("\t output(HB) = ");
  Serial.println(outputValue1);
  Serial.print("\t HB = ");
  Serial.print(sensorValue1);
  // wait 500 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(500);
  curSensoredTime = millis();
```

```
    // Read from sensor
  if(curSensoredTime - prevSensoredTime > SENSOR_READ_INTERVAL) {
    readFromSensor();  // Read from sensor
    prevSensoredTime = curSensoredTime;
      // Send buffer data to remote
    if(iAccelIndex >= ACCEL_BUFFER_COUNT - 3) {
      sendToRemote();
      initBuffer();
      Serial.println("------------ Send 20 accel data to remote");
    }
  }
}
/**************************************************
 * BT Transaction
 **************************************************/
void sendToRemote() {
  // Write`m  gabage bytes
  BTSerial.write( "accel" );
  // Write accel data
  BTSerial.write( (char*)aAccelBuffer );
  // Flush buffer
  //BTSerial.flush();
}
/**************************************************
 * Read data from sensor and save it
 **************************************************/
void readFromSensor() {
  int error;
  double dT;
  accel_t_gyro_union accel_t_gyro;
```

```
  error = MPU6050_read (MPU6050_ACCEL_XOUT_H, (uint8_t *)
&accel_t_gyro, sizeof(accel_t_gyro));
 if(error != 0) {
  Serial.print(F("Read accel, temp and gyro, error = "));
  Serial.println(error,DEC);
 }// Swap all high and low bytes.
 // After this, the registers values are swapped,
 // so the structure name like x_accel_l does no
 // longer contain the lower byte.
 uint8_t swap;
 #define SWAP(x,y) swap = x; x = y; y = swap
 SWAP (accel_t_gyro.reg.x_accel_h, accel_t_gyro.reg.x_accel_l);
 SWAP (accel_t_gyro.reg.y_accel_h, accel_t_gyro.reg.y_accel_l);
 SWAP (accel_t_gyro.reg.z_accel_h, accel_t_gyro.reg.z_accel_l);
 SWAP (accel_t_gyro.reg.t_h, accel_t_gyro.reg.t_l);
 SWAP (accel_t_gyro.reg.x_gyro_h, accel_t_gyro.reg.x_gyro_l);
 SWAP (accel_t_gyro.reg.y_gyro_h, accel_t_gyro.reg.y_gyro_l);
 SWAP (accel_t_gyro.reg.z_gyro_h, accel_t_gyro.reg.z_gyro_l);
 // Print the raw acceleration values
 Serial.print(F("accel x,y,z: "));
 Serial.print(accel_t_gyro.value.x_accel, DEC);
 Serial.print(F(", "));
 Serial.print(accel_t_gyro.value.y_accel, DEC);
 Serial.print(F(", "));
 Serial.print(accel_t_gyro.value.z_accel, DEC);
 Serial.print(F(", at "));
 Serial.print(iAccelIndex);
 Serial.println(F(""));
 if(iAccelIndex < ACCEL_BUFFER_COUNT && iAccelIndex > 1) {
  int tempX = accel_t_gyro.value.x_accel;
```

```
int tempY = accel_t_gyro.value.y_accel;
int tempZ = accel_t_gyro.value.z_accel;
/*
// Check min, max value
if(tempX > 16380) tempX = 16380;
if(tempY > 16380) tempY = 16380;
if(tempZ > 16380) tempZ = 16380;
  if(tempX < -16380) tempX = -16380;
if(tempY < -16380) tempY = -16380;
if(tempZ < -16380) tempZ = -16380;
// We dont use negative value
tempX += 16380;
tempY += 16380;
tempZ += 16380;
*/
char temp = (char)(tempX >> 8);
if(temp == 0x00)
  temp = 0x7f;
aAccelBuffer[iAccelIndex] = temp;
iAccelIndex++;
temp = (char)(tempX);
if(temp == 0x00)
  temp = 0x01;
aAccelBuffer[iAccelIndex] = temp;
iAccelIndex++;
temp = (char)(tempY >> 8);
if(temp == 0x00)
  temp = 0x7f;
aAccelBuffer[iAccelIndex] = temp;
iAccelIndex++;
```

```
    temp = (char)(tempY);
    if(temp == 0x00)
      temp = 0x01;
    aAccelBuffer[iAccelIndex] = temp;
    iAccelIndex++;
    temp = (char)(tempZ >> 8);
    if(temp == 0x00)
      temp = 0x7f;
    aAccelBuffer[iAccelIndex] = temp;
    iAccelIndex++;
    temp = (char)(tempZ);
    if(temp == 0x00)
      temp = 0x01;
    aAccelBuffer[iAccelIndex] = temp;
    iAccelIndex++;
  }
  // The temperature sensor is -40 to +85 degrees Celsius.
  // It is a signed integer.
  // According to the datasheet:
  // 340 per degrees Celsius, -512 at 35 degrees.
  // At 0 degrees: -512 - (340 * 35) = -12412
//  Serial.print(F("temperature: "));
//  dT = ( (double) accel_t_gyro.value.temperature + 12412.0) / 340.0;
//  Serial.print(dT, 3);
//  Serial.print(F(" degrees Celsius"));
//  Serial.println(F(""));
// Print the raw gyro values.
//  Serial.print(F("gyro x,y,z : "));
//  Serial.print(accel_t_gyro.value.x_gyro, DEC);
//  Serial.print(F(", "));
```

```
//  Serial.print(accel_t_gyro.value.y_gyro, DEC);
//  Serial.print(F(", "));
//  Serial.print(accel_t_gyro.value.z_gyro, DEC);
//  Serial.println(F(""));
}
/**************************************************
 * MPU-6050 Sensor read/write
 *************************************************/
int MPU6050_read(int start, uint8_t *buffer, int size)
{
  int i, n, error;

  Wire.beginTransmission(MPU6050_I2C_ADDRESS);
   n = Wire.write(start);
  if (n != 1)
    return (-10);
    n = Wire.endTransmission(false); // hold the I2C-bus
  if (n != 0)
    return (n);
   // Third parameter is true: relase I2C-bus after data is read.
  Wire.requestFrom(MPU6050_I2C_ADDRESS, size, true);
  i = 0;
  while(Wire.available() && i<size)
  {
    buffer[i++]=Wire.read();
  }
  if ( i != size)
    return (-11);
  return (0); // return : no error
}int MPU6050_write(int start, const uint8_t *pData, int size)
```

```
{
  int n, error;
   Wire.beginTransmission(MPU6050_I2C_ADDRESS);
 n = Wire.write(start); // write the start address
  if (n != 1)
    return (-20);
 n = Wire.write(pData, size); // write data bytes
  if (n != size)
    return (-21);
    error = Wire.endTransmission(true); // release the I2C-bus
  if (error != 0)
    return (error);
  return (0); // return : no error
}
int MPU6050_write_reg(int reg, uint8_t data)
{
  int error;
  error = MPU6050_write(reg, &data, 1);
  return (error);
}
/**************************************************
 * Utilities
 **************************************************/
void initBuffer() {
  iAccelIndex = 2;
  for(int i=iAccelIndex; i<ACCEL_BUFFER_COUNT; i++) {
    aAccelBuffer[i] = 0x00;
  }
  aAccelBuffer[0] = 0xfe;
  aAccelBuffer[1] = 0xfd;
```

```
  aAccelBuffer[122] = 0xfd;
  aAccelBuffer[123] = 0xfe;
  aAccelBuffer[124] = 0x00;
}
```

Step 5: Android

The Android app contains of 4 parts – Android UI, Bluetooth manager, Algorithm section, background service.

If Arduino powers on and the pairing process with FitnessBand app is done, the board check the accelerometer data 20 times in every 1 second. And it transfer the data to the mobile device once a second. The accelerometer measures x axis / y axis / z axis values, so the band sends 60 values (20 times x 3 axis) of data to the device. The Android app receives the data during two seconds and finds out an interval that user's movement increase dramatically. The number of user's movement increase is user's a step count. The app calculate burned calories based on user's weight and steps, and accumulates data monthly, daily and hourly.

Security

Fitness tracking devices monitor heartbeats, measure steps, and tie into a larger ecosystem of goal setting, diet tracking, and other health activities. *Every Step You Fake* investigates the privacy and security. A use of variety of technical, policy, and legal methods to understand what data is being collected by fitness tracking devices and their associated mobile applications, what data is sent to remote servers, how the data is secured, with whom it may be shared, and how it might be used.

The key technical findings include:

- The fitness tracking devices emit persistent unique identifiers (Bluetooth Media Access Control address) that can expose their wearers to long-term tracking of their location when the device is not paired, and connected to, a mobile device
- Applications like Jawbone and Withings applications can be exploited to create fake fitness band records. Such fake records call into question the reliability of that fitness tracker data use in court cases and insurance programs.
- Applications like The Garmin Connect (iPhone and Android) and Withings Health Mate (Android) have security vulnerabilities that enable an unauthorized third-party to read, write, and delete user data

Garmin Connect does not employ basic data transmission security practices for its iOS or Android applications and consequently exposes fitness information to surveillance or tampering.

Results

Install the app, run it and pair the mobile device with FitnessBand to see the app successfully receives the data. The app has 3 tab menus.

Timeline : It gathers cumulative burned calorie data every hour. You can check how many calories you have burned hourly / daily / monthly.

Graph : It shows the graph that is drawn by the data sent from the accelerometer. You can see how 3-axis values alter.

Settings : You can configure the app setting here, and input your weight here. Some features will be updated soon.

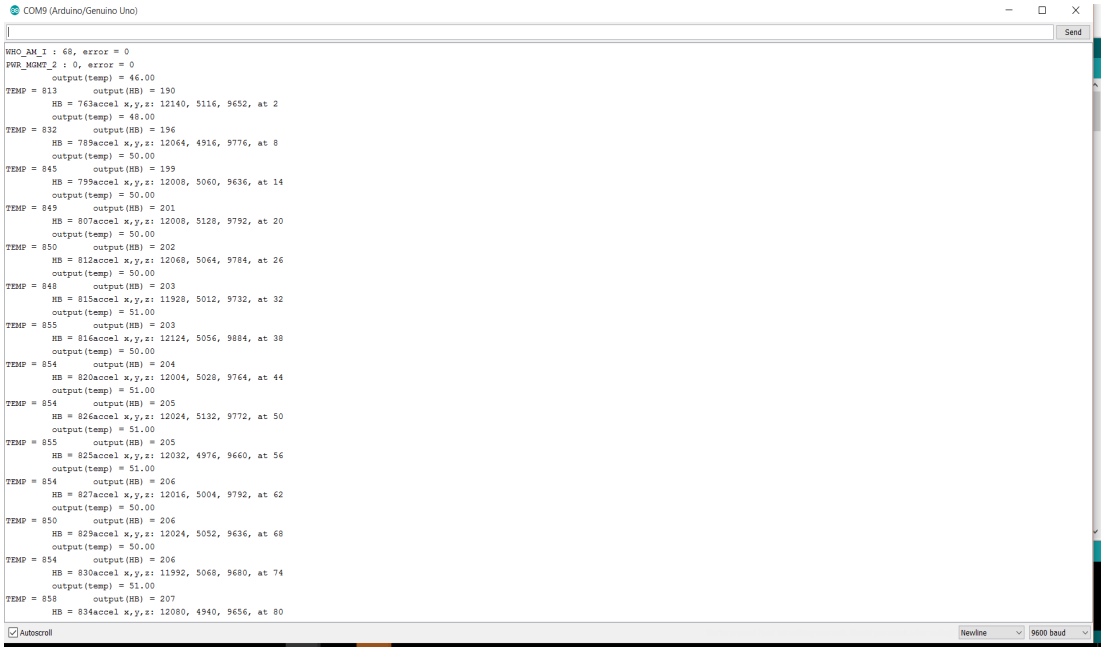
.

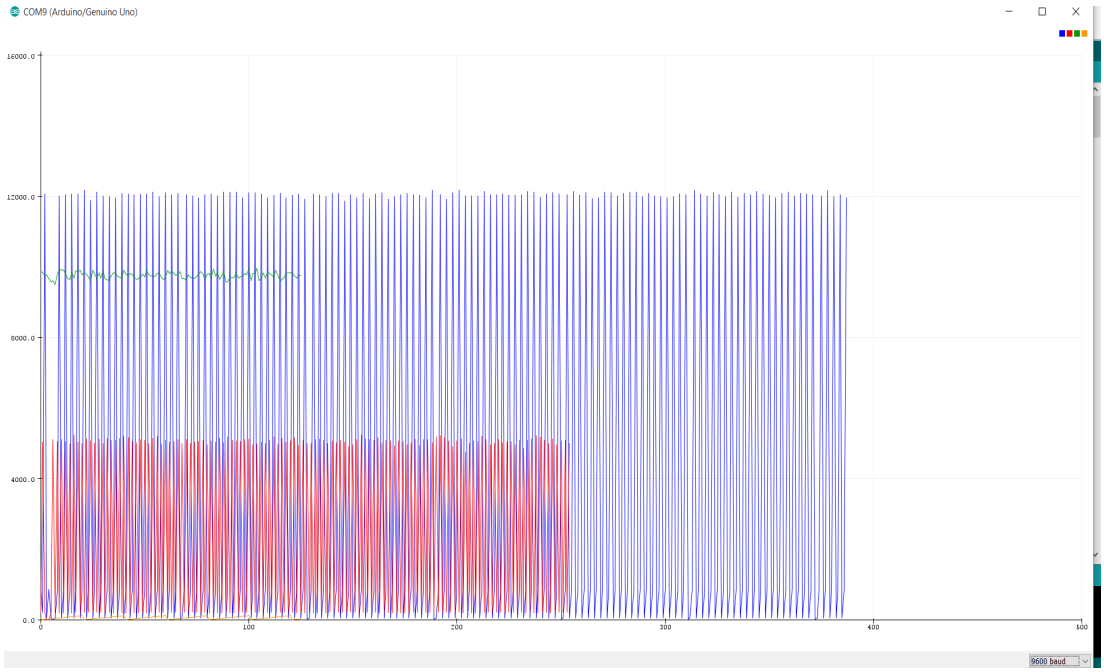
The specification of Fitness Band:

- Processor : ATmega328-3.3v(8MHz)
- 32KB Flash(2KB is shared for Bootloader), 2KB RAM, 1KB EEPROM
- Connected with Android exclusive app(supported over v.4.0).
- Calculating calories based on step count.
- Accumulating calorie data and displaying statistics in a monthly/daily/ hourly data
- Real-time check of the change of the 3-axis values measured by accelerometer
- Open source

Screenshots


오후 10:06
Retro Band
TIMELINE
GRAPH
SETTINGS
Calorie
Walks
2550
51
Activity report (Month)
Month
Bluetooth: connected HC-06


COM9 (Arduino/Genuino Uno)
Send
WHO_AM_I : 68, error = 0
PWR_MGMT_2 : 0, error = 0
output(temp) = 46.00
TEMP = 813 output(HB) = 190
HB = 763accel x,y,z: 12140, 5116, 9652, at 2
output(temp) = 48.00
TEMP = 832 output(HB) = 196
HB = 789accel x,y,z: 12064, 4916, 9776, at 8
output(temp) = 50.00
TEMP = 845 output(HB) = 199
HB = 799accel x,y,z: 12008, 5060, 9636, at 14
output(temp) = 50.00
TEMP = 849 output(HB) = 201
HB = 807accel x,y,z: 12008, 5128, 9792, at 20
output(temp) = 50.00
TEMP = 850 output(HB) = 202
HB = 812accel x,y,z: 12068, 5064, 9784, at 26
output(temp) = 50.00
TEMP = 848 output(HB) = 203
HB = 815accel x,y,z: 11928, 5012, 9732, at 32
output(temp) = 51.00
TEMP = 855 output(HB) = 203
HB = 816accel x,y,z: 12124, 5056, 9884, at 38
output(temp) = 50.00
TEMP = 854 output(HB) = 204
HB = 820accel x,y,z: 12004, 5028, 9764, at 44
output(temp) = 51.00
TEMP = 854 output(HB) = 205
HB = 826accel x,y,z: 12024, 5132, 9772, at 50
output(temp) = 51.00
TEMP = 855 output(HB) = 205
HB = 825accel x,y,z: 12032, 4976, 9660, at 56
output(temp) = 51.00
TEMP = 854 output(HB) = 206
HB = 827accel x,y,z: 12016, 5004, 9792, at 62
output(temp) = 50.00
TEMP = 850 output(HB) = 206
HB = 829accel x,y,z: 12024, 5052, 9636, at 68
output(temp) = 50.00
TEMP = 854 output(HB) = 206
HB = 830accel x,y,z: 11992, 5068, 9680, at 74
output(temp) = 51.00
TEMP = 858 output(HB) = 207
HB = 834accel x,y,z: 12080, 4940, 9656, at 80
Autoscroll
Newline
9600 baud


COM9 (Arduino/Genuino Uno)
16000.0
12000.0
8000.0
4000.0
0.0
0
100
200
300
400
500
9600 baud

Conclusion

All the objectives that is sensing heart beat and showing graph, sensing temperature, counting number of steps, calories burned are fulfilled in the given project.

Proper use of Arduino is made and the hardware. Each of the result is properly been shown on the computer screen and the android application via Bluetooth.

Overall, the Fitness Band provides a reasonable level of privacy for user data, but it would prefer a design that provided valid users an easy-to-access method for acquiring the full set of data recorded by the device.

Prototype Cost-

- Arduino UNO + USB cable – Rs 520
- Accelerometer(MPU-6050) – Rs 200
- Bluetooth module(HC-05) – Rs 345
- Temperature sensor(LM35) – Rs 100
- Heart beat pulse sensor – Rs 430
- Male-to-male jumper wires – Rs 150

  Total- Rs 1745 (approx.)

References

- IIT Bombay, Tech Fest
- Instructable.com
- www.hackster.io