CPE 301 EMBEDDED SYSTEM DESIGN S 2015

TITLE: Accelerometer + Wi-Fi

GOAL:

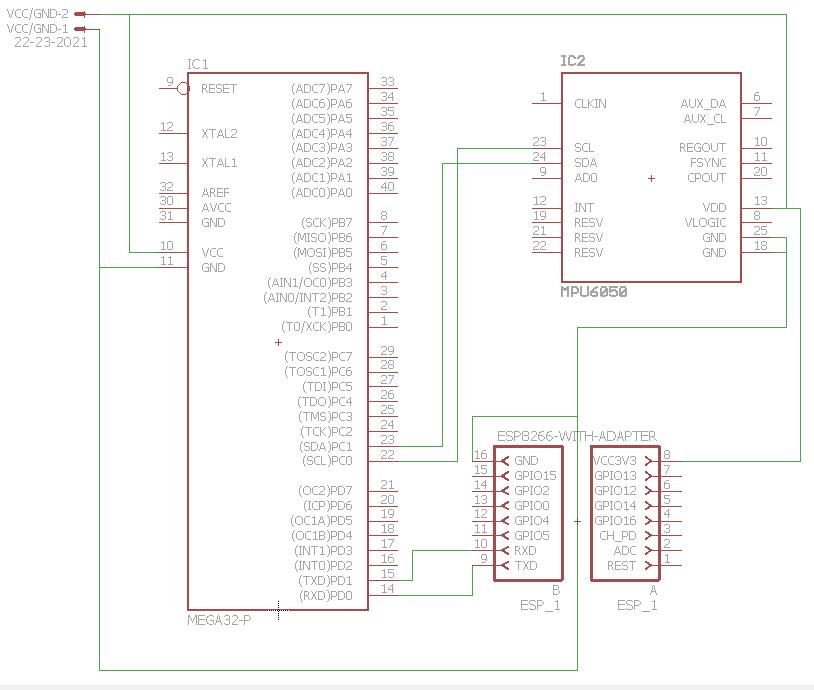
* Obtain data through accelerometer relating to the tilt of a surface
* Push that data to the cloud using a Wi-Fi Module to IOT
* Show the data in a meaningful way, due to Thingspeak limited mode of representations, when the line graph shows 100, surface is vertical. When 0 is displayed, surface is horizontal.

DELIVERABLES: Working circuit in which accelerometer sends data to cloud server whenever powered up and wifi is available. Code that is compliable and working.

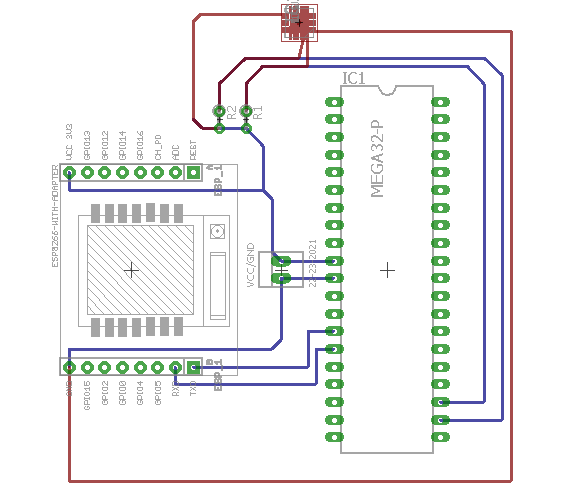
LITERATURE SURVEY:   
From minor applications, like knowing the direction in which to set a phone screen and mobile phone games, to potentially even keeping child seats in the up-right position in the event of an automobile accident or recognizing if a window has been opened. The accelerometer has become an important and unquestionable tool in localized and distant circuits. Phones use it locally to implement games or through Wi-Fi to know which webpage view to request based on the position the phone is held. A window or door being forced open can use its Wi-Fi function to alert the proper authorities.

COMPONENTS: Explain the main characteristics, interface, and limitation of the components used   
ATMEGA328P- Microcontroller, used to interface and connect the other components.  
ESP8266 – Wi-Fi Module; interfaced using USART to Atmega328P. Flashed with AT Firmware and pushed desired data to ThingSpeak IOT. The usage of ThingSpeak limits rate of pushes and display of that information.  
6 DOF Gyro, Accelerometer – Accelerometer used, XY and Z components used to detect shift in tilt of a surface. Interfaced using I2C to Atmega328P.   
FTDI chip – Used to interface with computer during flashing of ESP8266. Afterwards, it was used to check correct connection and outputs of ATmega and ESP8622 or ATmega and Accelerometer.

SCHEMATICS:



INITIAL PCB



IMPLEMENTATION: Based on the Project presentation materials and a quick literature survey - explain implementation details of the project. Make bullet points

* First thing is to flash ESP6288 with AT Firmware. This is done connecting the module directly to a PC using an FTDI chip. Firmware is installed using a flashing tool. From here, some settings are set. These include connection to wifi network.
* The goal was to identify if a surface, which would mimic a phone, was in a vertical or horizontal position. After trying many ideas and recording various outputs, it was noticeable that each position is basically rotating around one axis. So checking to see if the x or y axis is constant would reveal which position the surface was in.
* After proper libraries were implemented and coding of ATmega328P, ESP6288 is connected to RX/TX pins and interfaced using USART. Accelerometer is connected to the SCL and SDA with an interface using I2C. For testing purposes, an FTDI chip was connected to the USART as well to see correctness of outputs
* After proper connecting and power is applied, data from sensor is sent to microcontroller at every set interval.
* From here, the data and proper AT codes are sent to the Wi-Fi module which sends data to IOT. This turned out to be about a 25 second interval due to the connection and sending delay.
* IOT reflects data through a proper chart or display.
* A shift in tilt of the sensor is reflected by either a positive jump or negative jump, due to the direction of tilt. In the final implementation, a “100” being displayed on the IOT chart is a vertical stance, or portrait mode, and a “0” is a horizontal stance, or landscape mode.

SNAPSHOTS/SCREENSHOTS\*: (only links - do not embed images or videos in the document)   
1. Show snapshots/video of demo (IOT/BLE/VISUALIZATION).   
<https://www.youtube.com/watch?v=U8aB8dhnCCA>

CODE: (with comments)

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#include <util/twi.h>

#include "i2cmaster.c"

#include <math.h>

#include <string.h>

#define MPU60501 0xD0 // (0x68 << 1) I2C slave address

unsigned char ret; // return value

char ATs[110]; // Array for AT codes, needs to be long enought to fit everything

//char outs[50];

//char test[8] = "testing\n";

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

void usart\_init (void)

{

//synchronous usart, transmit 8-bit data

UCSR0C = ((1<<UCSZ01)|(1<<UCSZ00)|(1<<UMSEL00));

//115.2K Baud Rate from 8MHz clock

UBRR0H = 0x00;

UBRR0L = 0x22;

UCSR0B = (1<<TXEN0); //enable transmitter

}

//moved to i2cmaster for testing in there

/\*void USART\_tx\_string (char \*data)

{

while((\*data!='\0')){

while(!(UCSR0A&(1<<UDRE0))); //wait until transmit register is empty

UDR0 = \*data;

data++;

}

}\*/

void MPU6050\_writereg(*uint8\_t* accel, *uint8\_t* reg, *uint8\_t* val)

{

i2c\_start(accel+I2C\_WRITE);

i2c\_write(reg); // go to register e.g. 106 user control

i2c\_write(val); // set value e.g. to 0100 0000 FIFO enable

i2c\_stop(); // set stop condition = release bus

}

*uint16\_t* MPU6050\_readreg(*uint8\_t* accel, *uint8\_t* reg)//read unsigned 16 bits

{

i2c\_start\_wait(accel+I2C\_WRITE); // set device address and write mode

i2c\_write(reg); // ACCEL\_OUT

i2c\_rep\_start(accel+I2C\_READ); // set device address and read mode

int raw = i2c\_readAck(); // read one intermediate byte

raw = (raw<<8) | i2c\_readNak(); // read last byte

i2c\_stop();

return raw;

}

*int16\_t* MPU6050\_signed\_readreg(*uint8\_t* accel, *uint8\_t* reg)//read signed 16 bits

{

i2c\_start\_wait(accel+I2C\_WRITE); // set device address and write mode

i2c\_write(reg); // ACCEL\_OUT

i2c\_rep\_start(accel+I2C\_READ); // set device address and read mode

char raw1 = i2c\_readAck(); // read one intermediate byte

*int16\_t* raw2 = (raw1<<8) | i2c\_readNak(); // read last byte

i2c\_stop();

return raw2;

}

void Init\_MPU6050(*uint8\_t* accel)

{

ret = i2c\_start(accel+I2C\_WRITE); // set device address and write mode

if ( ret )//== "1" )

{

//snprintf(outs,sizeof(outs),"failed to issue start condition \n\r");

//USART\_tx\_string(outs);

i2c\_stop();

}

else

{

/\* issuing start condition ok, device accessible \*/

MPU6050\_writereg(accel, 0x6B, 0x00); // reg 107 set value to 0000 0000 and wake up sensor

MPU6050\_writereg(accel, 0x19, 0x07); // reg 25 sample rate divider set value to 0000 1000 for 1000 Hz

MPU6050\_writereg(accel, 0x1C, 0x18); // reg 28 acceleration configuration set value to 0001 1000 for 16g

//MPU6050\_writereg(accel, 0x1B, 0x18); // set gyro config value at

MPU6050\_writereg(accel, 0x23, 0xF8); // reg 35 FIFO enable set value to 1111 1000 for all sensors not slave

MPU6050\_writereg(accel, 0x37, 0x10); // reg 55 interrupt configuration set value to 0001 0000 for logic level high and read clear

MPU6050\_writereg(accel, 0x38, 0x01); // reg 56 interrupt enable set value to 0000 0001 data ready creates interrupt

MPU6050\_writereg(accel, 0x6A, 0x40); // reg 106 user control set value to 0100 0000 FIFO enable

//snprintf(outs,sizeof(outs),"done start \n\r");

//USART\_tx\_string(outs);

}

i2c\_stop();

}

int main(){

DDRD = 0xF0;

DDRC = 0x00;

int i=0;

int wifiout = -1; //value to display on chart in Thingspeak.

*int16\_t* xa1, ya1, za1;

//gyro

/\*int16\_t gyrox, gyroy, gyroz;

int16\_t gix = 0; //45

int16\_t giy = 0; //47

int16\_t giz = 0; //49\*/

//declare average calibrated accelerometer values

//initialize calibarition values

//declare accelerometer value strings

usart\_init();//initialize usart

//USART\_tx\_string(test);

i2c\_init(); // init I2C interface

*\_delay\_ms*(2000); // Wait for 200 ms.

Init\_MPU6050(MPU60501); // sensor init

*\_delay\_ms*(2000); // Wait for 200 ms.

//snprintf(outs,sizeof(outs),"6050 initialized \n\r");

//USART\_tx\_string(outs);

//resetting and setting up modes of wifi module

*\_delay\_ms*(1000);

*snprintf*(ATs,sizeof(ATs), "AT+RST\r");

USART\_tx\_string(ATs);

*\_delay\_ms*(2000);

*snprintf*(ATs,sizeof(ATs), "AT+CIPMODE=0\r");

USART\_tx\_string(ATs);

*\_delay\_ms*(1000);

*snprintf*(ATs,sizeof(ATs), "AT+CIPMUX=1\r");

USART\_tx\_string(ATs);

*\_delay\_ms*(2000);

//Start infinite loop

while(1){

//grab 3 values, average, and divide by MSB

//now multiplied by 2 because doubles dont want to print. This way at lease we get usable values.

// read raw X acceleration from fifo

xa1 = MPU6050\_signed\_readreg(MPU60501,0x3B)+MPU6050\_signed\_readreg(MPU60501,0x3B)+MPU6050\_signed\_readreg(MPU60501,0x3B);

xa1 = (2.00\*(xa1/3.00))/2048.00;

// read raw Y acceleration from fifo

ya1 = MPU6050\_signed\_readreg(MPU60501,0x3D)+MPU6050\_signed\_readreg(MPU60501,0x3D)+MPU6050\_signed\_readreg(MPU60501,0x3D);

ya1 = (2.00\*(ya1/3.00))/2048.00;

// read raw Z acceleration from fifo

za1 = MPU6050\_signed\_readreg(MPU60501,0x3F)+MPU6050\_signed\_readreg(MPU60501,0x3F)+MPU6050\_signed\_readreg(MPU60501,0x3F);

za1 = (2.00\*(za1/3.00))/2048.00;

if ((za1 == 1) && (xa1 == 0) && (ya1 == 0))

{

wifiout = wifiout; //while flat, keep last mode

}

else if (ya1 == 0)

{

wifiout = 0; //application: to see if phone is in landscape mode, vertical

}

else if (xa1 == 0)

{

wifiout = 100; //application, to see if phone is in portrait mode, horizontal

}

//Send to WIFI MODULE

//every 25 secs

//send code

*\_delay\_ms*(100);

// AT Code to connect to thingspeak IOT

*snprintf*(ATs,sizeof(ATs), "AT+CIPSTART=0,\"TCP\",\"api.thingspeak.com\",80\r");

USART\_tx\_string(ATs);

*\_delay\_ms*(2000);

*snprintf*(ATs,sizeof(ATs), "AT+CIPSEND=0,125\r");

USART\_tx\_string(ATs);

*\_delay\_ms*(1000);

// AT Code containing KEY and value to be displayed.

*snprintf*(ATs,sizeof(ATs), "GET https://api.thingspeak.com/update?api\_key=KX3UE6FR4YHJOFL1&field1=%i HTTP/1.0\n\r", wifiout);

USART\_tx\_string(ATs);

*snprintf*(ATs,sizeof(ATs), "\n\r");

while (i < 22)

{

i++; //needed to push data through.

USART\_tx\_string(ATs);

*\_delay\_ms*(1000);

}

i=0;

*\_delay\_ms*(100);

}

return 0;

}

REFERENCES  
1. <http://www2.usfirst.org/2005comp/Manuals/Acceler1.pdf>

T.I Accelerometer informational slides; used to understand raw data output from accelerometer.

2. <http://homepage.hispeed.ch/peterfleury/doxygen/avr-gcc-libraries/group__pfleury__ic2master.html>

Peter Fleur’s I2C library; used to understand how I2C works and how to use it to connect to the accelerometer.

3. ATmega328P Datasheet

4. MPU 6050 Datasheet and Product Specification Sheet

5. <https://nurdspace.nl/ESP8266#AT_Commands>

Learn the AT commands

6. <https://www.youtube.com/watch?v=qU76yWHeQuw>

Understand the connections, theory, and implementation of the ESP8266 Wifi Module

Name: Emmanuel Rodriguez Lopez