CPE 301 EMBEDDED SYSTEM DESIGN S 2015

TITLE: IMU WIFI IOT

GOAL:

* Read data from IMU6050 to ATMega328P
* Send data from ATMega328P to ESP8266
* Have ESP8266 module upload data to ThingSpeak cloud service

DELIVERABLES:

An IoT device capable of uploading IMU gathered data via WiFi

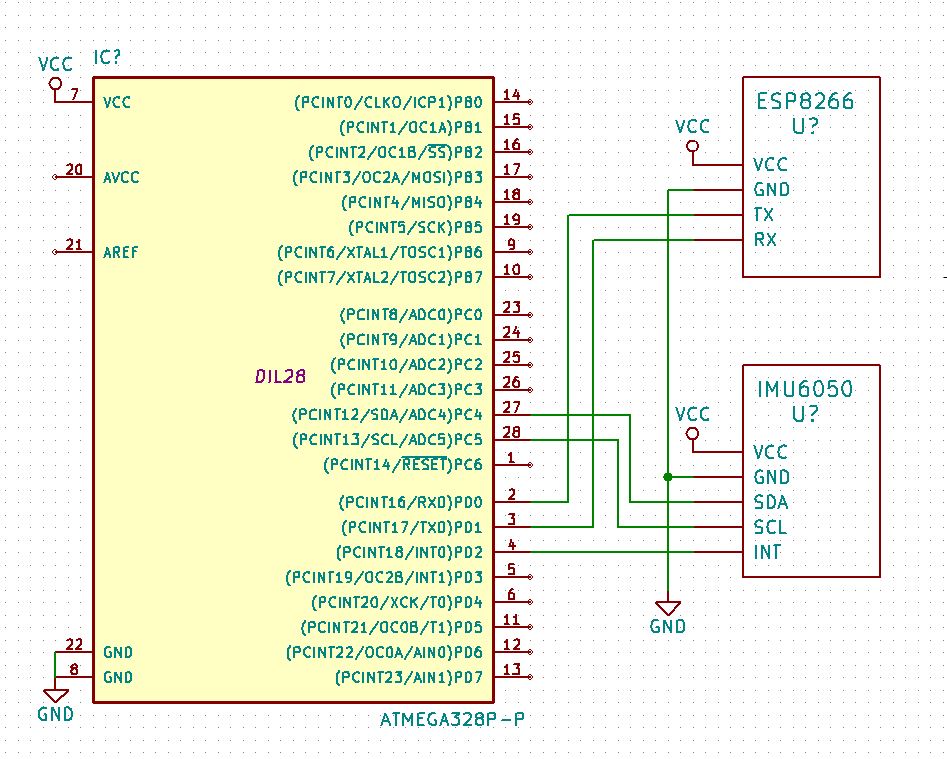
LITERATURE SURVEY:

Allows for gathering of data that can be accessed and analyzed by anyone on the web.

COMPONENTS:

* ATMega328P – main controller of device, reads data from sensor and pushes the data to a Wifi module
* IMU6050 – Gathers data, utilizes I2C communication standard to send data to ATMega328P
* ESP8266 – used to upload gathered data to ThingSpeak cloud service via WiFi, communicates with ATMega328P via UART

SCHEMATICS:



INITIAL PCB\*: (work in progress)

IIMPLEMENTATION:

* Implemented on bread board

SNAPSHOTS/SCREENSHOTS\*:

<http://imgur.com/e0RSq6i>

CODE: (Changes highlighted in yellow)

/\* Brandon Wade\*/

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#include <util/twi.h>

#include "i2cmaster.h"

#include <math.h>

#include <string.h>

#include <stdlib.h>

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

unsigned char ret; // return value

char outs[50];

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

void usart\_init (void)

{

//synchronous usart, transmit 8-bit data

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

//9600 Baud Rate from 8MHz clock

UBRR0H = 0;

UBRR0L = 33;

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

}

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 )

{

*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, 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(){

*int16\_t* xi1 = 0;

*int16\_t* yi1 = 0;

*int16\_t* zi1 = 0;

float xa1, ya1, za1;

int x0,x1,x2,y0,y1,y2,z0,z1,z2;

DDRD = 0xF0;

DDRC = 0x00;

//declare average calibrated accelerometer values

//initialize calibarition values

//declare accelerometer value strings

usart\_init();//initialize usart

i2c\_init(); // init I2C interface

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

Init\_MPU6050(MPU60501); // sensor init

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

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

USART\_tx\_string(outs);

for(int i = 0; i<10; i++)//get values for initial calibration

{

// read raw X acceleration from fifo

xi1 += MPU6050\_signed\_readreg(MPU60501,0x3B);

// read raw Y acceleration from fifo

yi1 += MPU6050\_signed\_readreg(MPU60501,0x3D);

// read raw Z acceleration from fifo

zi1 += MPU6050\_signed\_readreg(MPU60501,0x3F);

}

//average values for calibration

xi1 = xi1/10;

yi1 = yi1/10;

zi1 = zi1/10;

//Start infinite loop

while(1){

//grab 3 values, average, subtract calibration value, and divide by MSB

// read raw X acceleration from fifo

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

xa1 = ((xa1/3)-xi1)/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 = ((ya1/3)-yi1)/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 = ((za1/3)-zi1)/2048.00;

//convert doubles to printable strings

x0 = (int)xa1%10; // ones place

x1 = *abs*((int)(xa1 \*10)%10); // tenths place

x2 = *abs*((int)(xa1 \*100)%10); // hundredths place

y0 = (int)ya1%10;

y1 = *abs*((int)(ya1 \*10)%10);

y2 = *abs*((int)(ya1 \*100)%10);

z0 = (int)za1%10;

z1 = *abs*((int)(za1 \*10)%10);

z2 = *abs*((int)(za1 \*100)%10);

//print out the values

*snprintf*(outs,sizeof(outs),"x: %i.%i%i, y: %i.%i%i, z: %i.%i%i\n\r",x0,x1,x2,y0,y1,y2,z0,z1,z2); // print all data on one line

USART\_tx\_string(outs); // tx

*\_delay\_ms*(500); // wait after each print

}

return 0;

}

REFERENCE:

MPU6050\_help files sent

Name: Brandon Wade Page 1/1