Andrew Wells

TITLE: MPU-6050 Acceleration and Gyroscopic Data to Cloud

GOAL:

* Connect ESP8266 to a cloud IOT service
* Successfully connect ATmega328P to ESP8266 with UART
* Send commands from ATmega328P to ESP8266 using AT format
* Receive values from MPU-6050 6DOF sensor
* Transmit values from MPU-6050 to cloud

DELIVERABLES:

The project is intended to deliver sensor data to the cloud. With minor adjustments to some specific variables the code should be portable to any sensor attached to the ATmega326P.

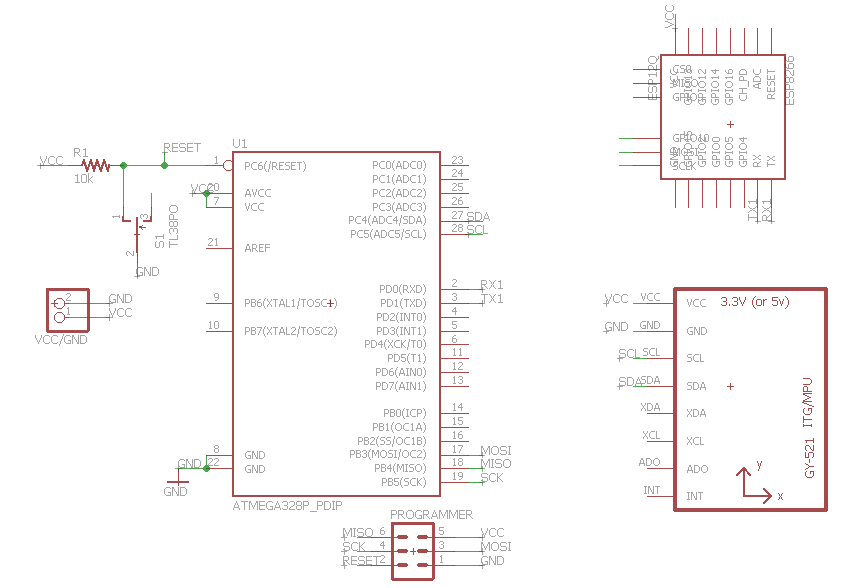
LITERATURE SURVEY:

Used I2C to communicate between the Xplained mini and the MPU-6050. Using UART on the Xplained mini to transmit from data received from the sensor to the ESP-8266. Used AT commands to direct the ESP-8266 to pass the data to the cloud. The cloud service that was chosen to transmit the data to was Thingspeak.com. This cloud service is able to receive data at a rate of once every 15 seconds.

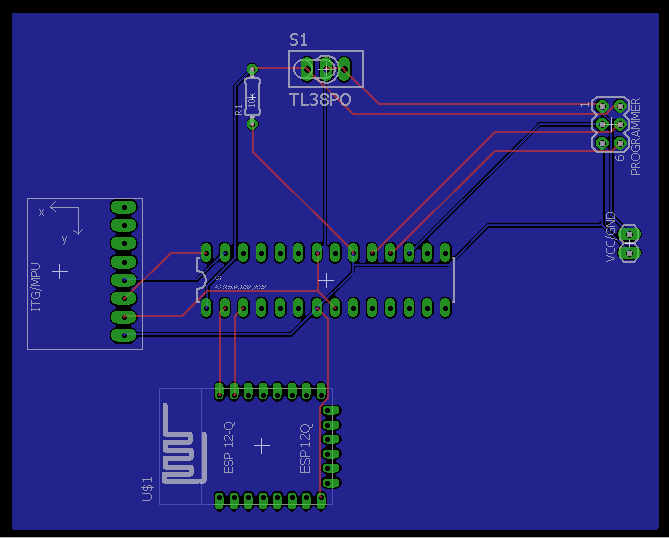
COMPONENTS:

Using the Xplained Mini with the ATmega328P chip I was able to connect the MPU-6050 and ESP8266. Using Atmel Studio, I programmed the ATmega328P using C. For this project there were no limitations while using the chip, only my ability to properly program using C. The MPU-6050 was connected using I2C. Using the header files at <https://github.com/YifanJiangPolyU/MPU6050>, I was able to interface and receive acceleration data from the sensor. The primary limitation of this device is the speed at which data can be received and processed by the micro controller. The ESP8266 was needed to connect to the Wifi and transmit the sensor data to the cloud. Commands were transmitted using UART over a serial connection using AT commands. A limit of the AT firmware is there is not a command to connect using https. Also there is not a way to connect to a Wifi signal that requires a username and password, can only connect to a SSID that requires a simple password. The biggest limitation to this project is the upload speed available using a free IOT service. Using Thinkspeak.com to receive the data limits uploads to every 15 seconds.

SCHEMATICS: (exception - include image)



INITIAL PCB\*: (exception - include image)



IMPLEMENTATION:

* Flashed AT command firmware onto ESP-8266
* Read raw data from MPU-6050 using I2C
* Calculate true acceleration value
* Calculate true gyroscopic value
* Transmit data over UART from Xplained mini to ESP-8266
* Used the command AT+CIPSTART to connect to Thingspeak.com
* Used the command AT+CIPSEND to transmit with upload string

SNAPSHOTS/SCREENSHOTS\*: (only links - do not embed images or videos in the document)

Show snapshots/video of component implementation.

Show snapshots/video of demo (IOT/BLE/VISUALIZATION).

YouTube Link:

<https://youtu.be/0cHQnD4WA8E>

All other images are located in my Github repository at:

<https://github.com/Wellsa15/wellsa_unlv>

Thingspeak Public Channel:

<https://thingspeak.com/channels/110409>

CODE: (with comments) [\*final code]

#define *F\_CPU* 8000000UL // Set clock to 8MHz

#define UBRR\_115200 8 //For 8Mhz

#include <avr/io.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include "mpu6050.h"

#include "mpu6050\_reg.h"

#include "i2c.h"

void USARTstart(unsigned int ubrr); // Function to initialize the USART port

void USART\_tx\_string(char \*data); // Function to transmit the value over the serial port

char Xout[10]; // output buffer for Acceleration X-axis

char Yout[10]; // output buffer for Acceleration y-axis

char Zout[10];// output buffer for Acceleration z-axis

char GyroXout[10]; // output buffer for Gyro X axis

char GyroYout[10]; // output buffer for Gyro Y axis

char GyroZout[10]; // output buffer for Gyro Z axis

char DATA\_STR[120]; // buffer for string that sends data to cloud

char WIFI\_COMMAND[] = "AT+CWJAP=\"XXX\",\"xxx\"\r\n\0"; // String to connect to Wifi

char CONNECT\_COMMAND[] = "AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n\0";// Command to connect to Thingspeak.com

char SEND\_COMMAND[20]; // Buffer for string to tell wifi how many characters will be sent

// Wifi connect: AT+CWJAP="SSID", "PASSWORD"

// Connect to Thinkspeak: AT+CIPSTART= "TCP","%s(IP)" , 80 \*Add \ in front of " for it to be read properly

// Send data: AT+CIPSEND=(string.length())

// Data string to send: "GET /update?api\_key=%s&field1=(sensor value)\r\n\r\n",APIKEY,Sensor)

int main(void)

{

int counter; // Counter for number of characters to be sent

*uint8\_t* ret;

*int16\_t* accel\_buff[3];// buffer to hold acceleration raw data

*int16\_t* gyro\_buff[3]; // buffer to hold gyroscope raw data

double accelX, accelY, accelZ; // Holds calculated acceleration values

double gyroX, gyroY, gyroZ; // Holds calculated gyroscope values

sei (); // enable global interrupts

USARTstart(UBRR\_115200); // initialize the USART port

i2c\_init();// initialize I2C

// Build command connect to Wifi and transmit

// strcat(WIFI\_COMMAND,SSID);

// strcat(WIFI\_COMMAND,PASSWORD);

// USART\_tx\_string(WIFI\_COMMAND);// Connect to Wifi

DDRB |= (1<<5);

//initialize & test MPU5060 availability

ret = i2c\_start(MPU6050\_ADDRESS+I2C\_WRITE);

if(~ret)

{

PORTB |= (1<<5);

*\_delay\_ms*(200);

PORTB &= ~(1<<5);

}

mpu6050\_init();// Initialize MPU

for (int b=0; b<10;b++) // Delay to give wifi time to connect and be ready

*\_delay\_ms*(1000);

while (1) // infinite loop to transmit the sensor value over serial port

{

counter = 0;

mpu6050\_read\_accel\_ALL(accel\_buff);// Read acceleration data

mpu6050\_read\_gyro\_ALL(gyro\_buff);// Read gyroscopic data

// acceleration (m/s^2)

accelX = accel\_buff[0]/16384.0; // Calculate Acceleration X-axis value

accelY = accel\_buff[1]/16384.0; // Calculate Acceleration Y-axis value

accelZ = accel\_buff[2]/16384.0; // Calculate Acceleration Z-axis value

//

gyroX = gyro\_buff[0]/32.8; // Calculate Gyroscopic X-axis value

gyroY = gyro\_buff[1]/32.8; // Calculate Gyroscopic Y-axis value

gyroZ = gyro\_buff[2]/32.8; // Calculate Gyroscopic Z-axis value

//Convert Acceleration X,Y, and Z values to Strings

if (accelX < 0)

*dtostrf*(accelX, 6, 3, Xout);

else

{

*dtostrf*(accelX, 6, 3, Xout);

*memmove*(Xout, Xout+1,sizeof(Xout));

}

if (accelY < 0)

*dtostrf*(accelY, 6, 3, Yout);

else

{

*dtostrf*(accelY, 6, 3, Yout);

*memmove*(Yout, Yout+1,sizeof(Yout));

}

if (accelZ < 0)

*dtostrf*(accelZ, 6, 3, Zout);

else

{

*dtostrf*(accelZ, 6, 3, Zout);

*memmove*(Zout, Zout+1,sizeof(Zout));

}

// Convert Gyroscopic X,Y, and Z values to strings

if (gyroX < 0)

*dtostrf*(gyroX, 6, 3, GyroXout);

else

{

*dtostrf*(gyroX, 6, 3, GyroXout);

*memmove*(GyroXout, GyroXout+1,sizeof(GyroXout));

}

if (gyroY < 0)

*dtostrf*(gyroY, 6, 3, GyroYout);

else

{

*dtostrf*(gyroY, 6, 3, GyroYout);

*memmove*(GyroYout, GyroYout+1,sizeof(GyroYout));

}

if (gyroZ < 0)

*dtostrf*(gyroZ, 6, 3, GyroZout);

else

{

*dtostrf*(gyroZ, 6, 3, GyroZout);

*memmove*(GyroZout, GyroZout+1,sizeof(GyroZout));

}

// Build Data string to send to cloud

*snprintf*(DATA\_STR, sizeof(DATA\_STR),"GET /update?api\_key=U6KEQ2CNJNAK85WX&field1=%s&field2=%s&field3=%s&field4=%s&field5=%s&field6=%s\r\n", Xout, Yout, Zout, GyroXout, GyroYout, GyroZout);

while (DATA\_STR[counter] != '\0') // Count how many characters in the output string

{

counter++;

}

// Build Command to Send the length of the string to follow

*snprintf*(SEND\_COMMAND, sizeof(SEND\_COMMAND),"AT+CIPSEND=%d\r\n",counter);

USART\_tx\_string(CONNECT\_COMMAND);// Connect to Thinkspeak.com

*\_delay\_ms*(400);

USART\_tx\_string(SEND\_COMMAND);// Transmit command to send data

*\_delay\_ms*(100);

USART\_tx\_string(DATA\_STR); // transmit data over serial connection

for (int i = 0; i<16; i++)// Wait 15sec to upload next dataset

*\_delay\_ms*(1000);

}

return 0;

}

void USARTstart(unsigned int ubrr)

{

UBRR0H = (unsigned char)(ubrr>>8);// set baud rate to 115200

UBRR0L = (unsigned char)ubrr;

UCSR0B |= (1<<RXEN0) | (1<<TXEN0); // Enable Transmit and Receive

UCSR0C |= (1<<UCSZ01) | (1<<UCSZ00); // Set Frame: 8bit, 1 Stop

return;

}

void USART\_tx\_string(char \*data)// transmit value function

{

while ((\*data != '\0'))// loop to send each character over the serial connection

{

while (!(UCSR0A & (1<<UDRE0)))// wait until transmit buffer is clear

{

}

UDR0 = \*data; // load character into transmit buffer

data++; // go to next character

}

}

REFERENCE:

* Class Slides
* <https://github.com/YifanJiangPolyU/MPU6050>
* Thingspeak.com Documentaion: <https://www.mathworks.com/help/thingspeak/?refresh=true>

PS: \* - can be omitted in the Pre-Final Report.