AUTOMOTIVE WHEEL ALIGNMENT SYSTEM

By

Michael Danek

Isaac Kousari

Final Report for ECE 445, Senior Design, Fall 2016

TA: Luke Wendt

7 December 2016

Project No. 48

Abstract

Modern automotive sensing systems are complex and address a wide variety of common maintenance issues. When issues affect human safety or have environmental repercussions, these sensing systems address the issue by notifying the user to have their vehicle serviced. As advanced as modern sensing systems are, one does not exist for vehicle wheel alignment. Our goal was to develop such a sensor system that could be integrated into new vehicles being developed by automobile manufacturers. This system would have the capability to determine a car's toe, camber, and caster with accuracy within ±.1° of these primary angles. At a cost of $140 for parts per car, this system costs substantially less over time than the average cost of getting a wheel alignment at a dealership, $180 [1]. This figure also does not include the time cost to the user for getting a wheel alignment.

Contents

1. Introduction

1.1 The Problem

Automobile efficiency is a prevalent topic today. As each automobile manufacturer strives to create state-of-the-art vehicles that are comfortable to drive and meet strict government-set carbon emission standards, one aspect of automobile performance has been historically neglected: wheel alignment. Poor wheel alignment causes a slew of issues, ranging from performance issues to more serious issues such as user safety.

Out-of-specification camber causes one side a tire to wear faster than the other [1], necessitating more frequent tire changes. If there is a camber mismatch between sides of a vehicle, the car will tend to pull toward the side with more positive camber while it is in motion. Out-of-specification caster causes issues with straight-line tracking [1]. In other words, a car with out-of-specification caster cannot be driven in a straight line without turning the steering wheel to some degree. Out-of-specification toe affects tires on both sides of the car to wear rapidly as rolling resistance is increased [1]. Toe can also affect thrust angle of the vehicle, meaning that the car will tend to turn in one direction, not allowing the steering wheel to be centered when travelling in a straight line. So, any state of poor wheel alignment results in unpredictable handling characteristics.

If left unchecked, these basic alignment angles, or primary angles, can reduce automobile efficiency in that they increase the rolling resistance of the tires, resulting in greater fuel expenditure and tire wear. Furthermore, a vehicle's suspension parts will also require premature replacement due to undesired force distribution while the vehicle is in motion. These side effects result in greater carbon emissions, whether it be directly through increased levels of fuel consumption or indirectly through increased levels of demand for automotive parts. As part manufacturers face increased demand for parts, more power will be used to manufacture parts, and any power not derived from renewable energy sources will result in greater carbon emissions.

1.2 Our Solution

Our goal is to develop an advanced sensor system capable of detecting wheel alignment issues before car parts become too worn out or hazardous to user safety. This system will function such that users can forego the time-consuming, costly process of taking their automobiles to an alignment shop or car dealership merely to have an alignment check performed. To achieve this goal, we designed a wireless, 5-node system of sensors intended to be integrated into new vehicles designs. Each sensor contains an accelerometer, to be mounted on a vehicle in a precise frame of reference so that wheel orientation data can be determined from relative accelerometer data. 4 of 5 sensors are mounted within wheel center caps, while the final sensor serves as a central reference point, mounted to the chassis of the vehicle.

Wheel alignment is measured in tenths or hundredths of degrees [3]. Given the sensitivity of the accelerometers used in our design, we chose to aim for a precision of ±.1° for each primary angle measurement. As of 2007, the average lengths of compact sedans and compact sport utility vehicles in America are 14.77 feet and 14.36 feet, respectively [3]. Since our central sensing unit and microprocessor are to be installed in a centralized location on the chassis, our goal to have sensors communicate with each other at distances of up to 10 feet away ensures that this system could function when installed in the average new car.

2 Design

In our design, we separated our system into four main blocks: sensors, control, power and communication. For our system, we knew that the design must be low power and compact to be used on vehicles, such that the accuracy remains high and the robustness is not compromised. It was decided that a centralized network of sensor nodes that collect data and then transmit the data to the center would produce the most efficient solution. This way, if the centralized node was placed in the physical "center" of the car, the device would have access to a stable and continuous power supply and thus have less of a lifetime issue than that of the network edge nodes. Edge nodes are placed on the center cap of wheels to have minimal external force applied. This produces a more robust system by reducing the stress due to acceleration from daily driving. Referring to (Figure 1), the system set up can be seen.

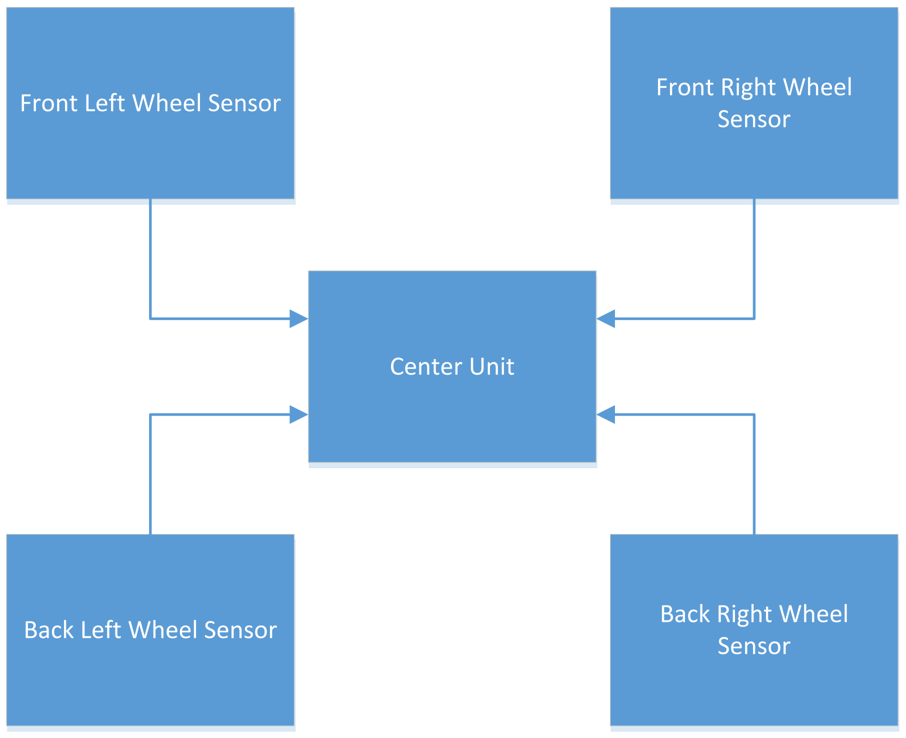

Figure 1: System Block Diagram

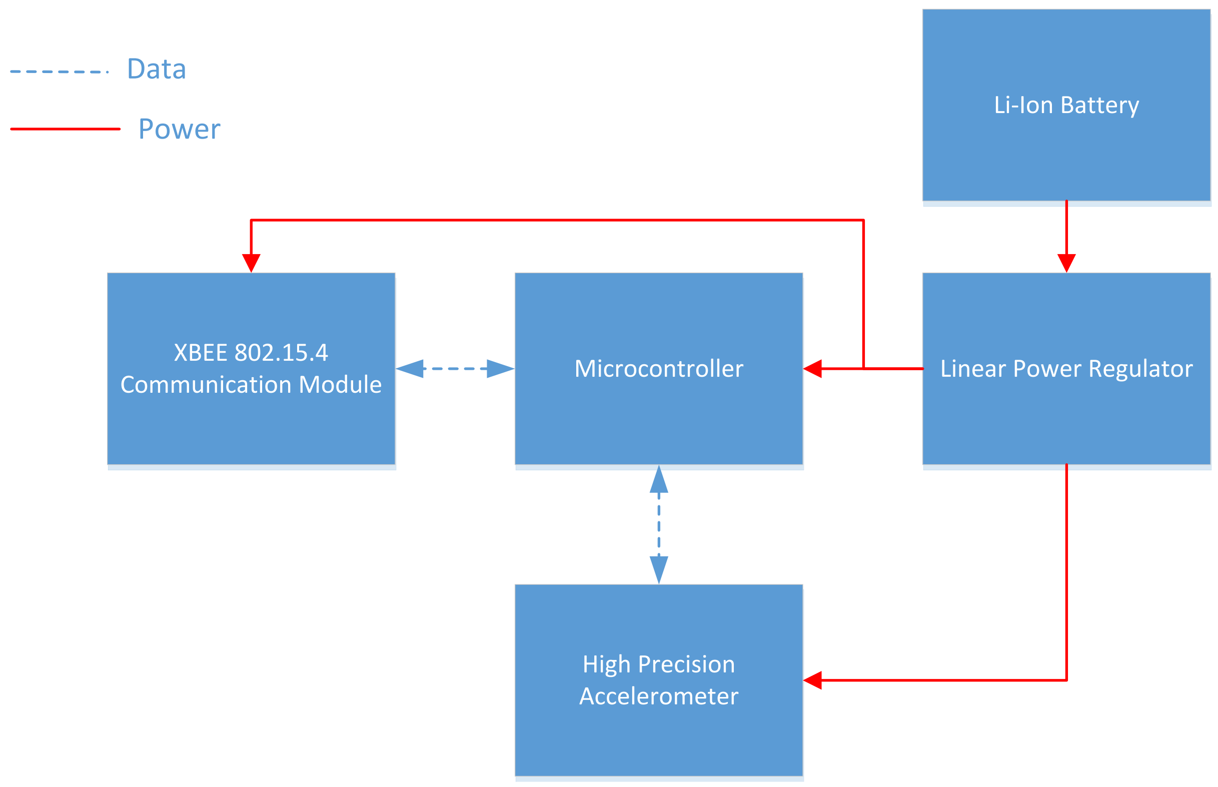
Data
Power
Li-Ion Battery
XBEE 802.15.4 Communication Module
Microcontroller
Linear Power Regulator
High Precision Accelerometer

Figure 2: Node Block Diagram

The general set up of each individual sensor is described above in (Figure 2); five of these sensors linked together in a centralized network forms the system.

2.1 Power System

Our power system is composed of two subcomponents: A Li-Ion battery and a linear regulator. The linear regulator is the Texas Instruments LP5912, it will be further described in its respective sub-block. The other component, the Li-Ion battery is an 18650 internally protected 3.7V 2600 mAh battery.

Table 1

Table 1: Power Consumption

Part	Voltage	Idle	Active
LP5912	3.3 V	10 uA	10 uA
ATMEGA328P MLF	3.3 V	1.2 mA	5.2 mA
LIS3DSH	3.3 V	7 uA	0.225 mA
XBee 802.15.4	3.3 V	55 mA	215 mA
Total		**185.52 mW**	**727.44 mW**

2.1.1 LP5912 Linear Regulator

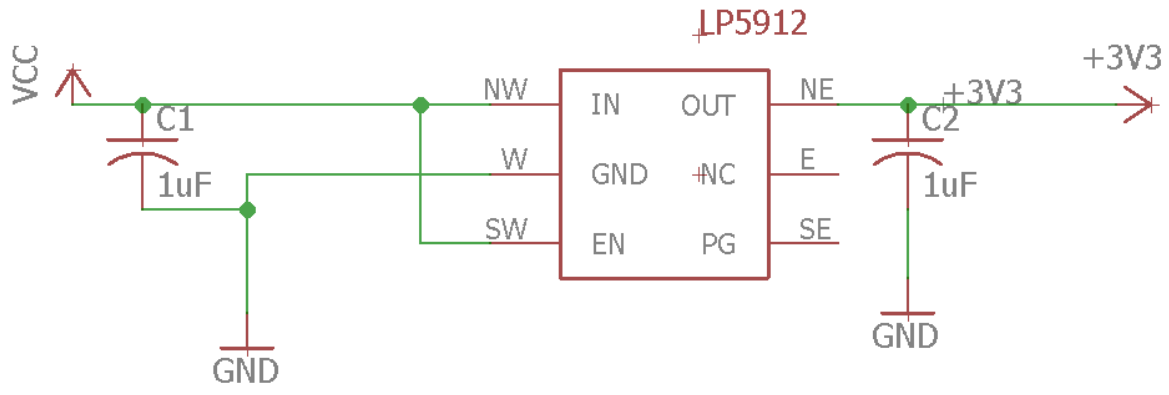

Figure 3: LP5912 LDO

The LP5912 is a linear regulator designed by Texas Instruments which is a LDO (Low Dropout) Regulator. The LP5912 was chosen because of both its input range and its output tolerance, it can accept voltages from -.3V to 7V and has an output tolerance range of +/- 2%. In addition, the PSRR is conservatively 75 dB at 1 kHz, which helps to reduce output voltage ripple. The leakage current is 10uA, which affects total power consumption, however given that the current is on the micro- scale versus the milli- scale which the rest of the system is under. Using capacitors on the input and output of the of the linear regulator provides a lowpass filter which eliminates high frequency noise. Although a resistor is not seen at the output of figure 3, the LDO has an intrinsic resistance therefore it will behave like a load. In addition, the rest of the circuit acts as a load to the output to complete the output lowpass filter.

2.1.2 Li-Ion Battery

The Lithium Ion battery that we chose for the system was a 3.7V, 2600 mAh,18650 model. The battery has a PCB that internally protects it from short circuit damages. If excessively charged, the PCB detects this and limits the damage done to the battery and regulates the output in addition the maximum over charge voltage which is 4.2V, is within the range of the LDO (7V).

2.2 Control System

All operations of the sensor are controlled by a single microcontroller that tells the sensors and the transceiver when to read or write data. Since the system is mostly in an idle state the microcontroller is not executing code and is in a no-op loop until it receives an outside command from the central node. The microcontroller then targets the sensors to turn on the sensors for the sampling period until the data collection is complete.

2.2.1 Microcontroller

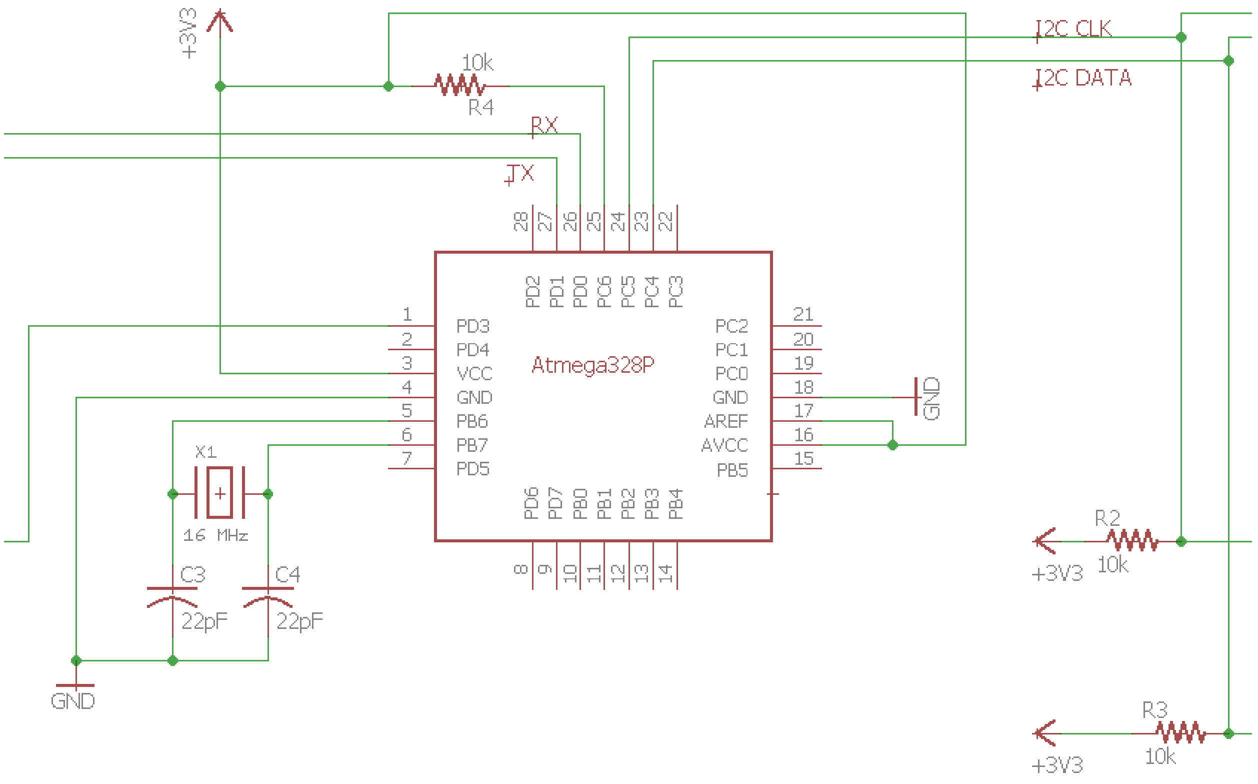

Figure 4: Atmega328P Microcontroller

The microcontroller that was used was the Atmega328P, it was selected because of its open support for development and decent power margins. The Atmega328P has a C compiler available for it, that can be used to develop code such that the device defined assembly language and port targeting can be avoided. Although the Atmega328P has an internal 8 MHz clock it supports external clocks, in our design we added an external 16 MHz clock. In addition, the device natively supports 2 wire communication over I2C which is used for communicating with the sensors.

2.3 Sensor System

2.3.1 Accelerometer

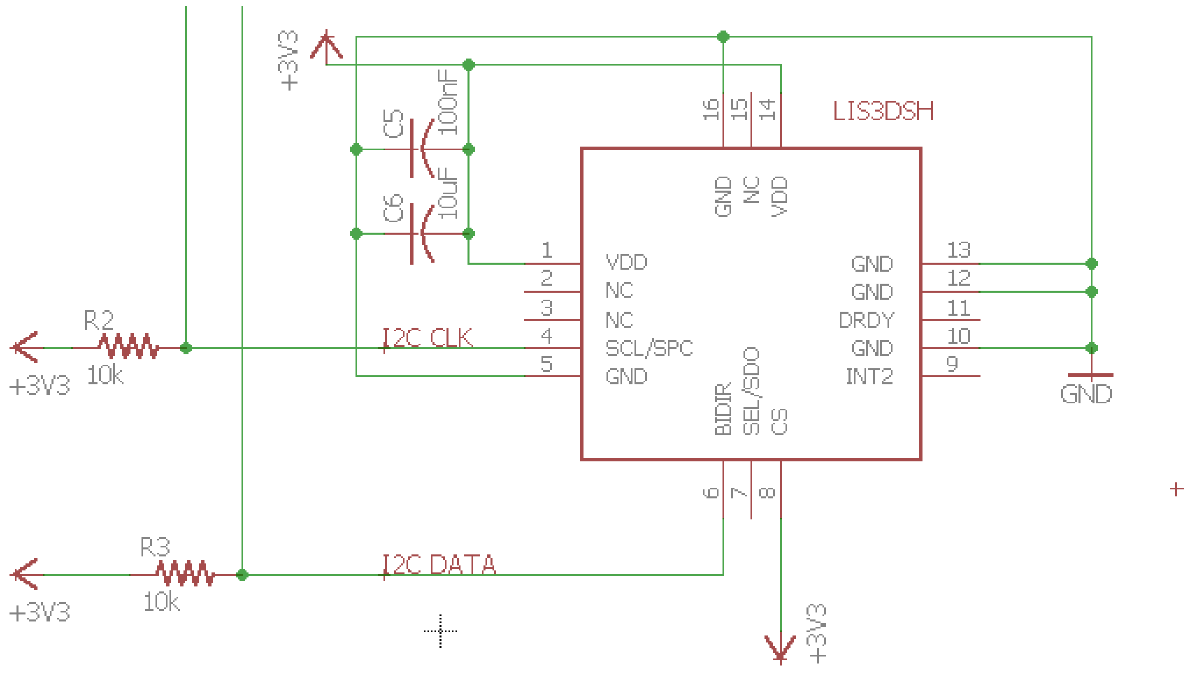

Figure 5: LIS3DSH 3 Axis Accelerometer

The device that we used for the gathering of acceleration data for calculating angle data was the LIS3DSH accelerometer. This device has a resolution of 16333 LSB/g this resolution allows for the precision of the device to be accurate to 0.1°. The device has support for I2C it has a simple communication protocol that allows for an easy interface with the microcontroller. The devices power characteristics allows the device to be in an idle, "powered down", state that uses minimal current. This allows the node to consume minimal power when it is in a wait state. Per (Figure 6), the noise density that affects the calculations are insignificant after 20 samples. This allows the calculations to be performed without additional hardware.

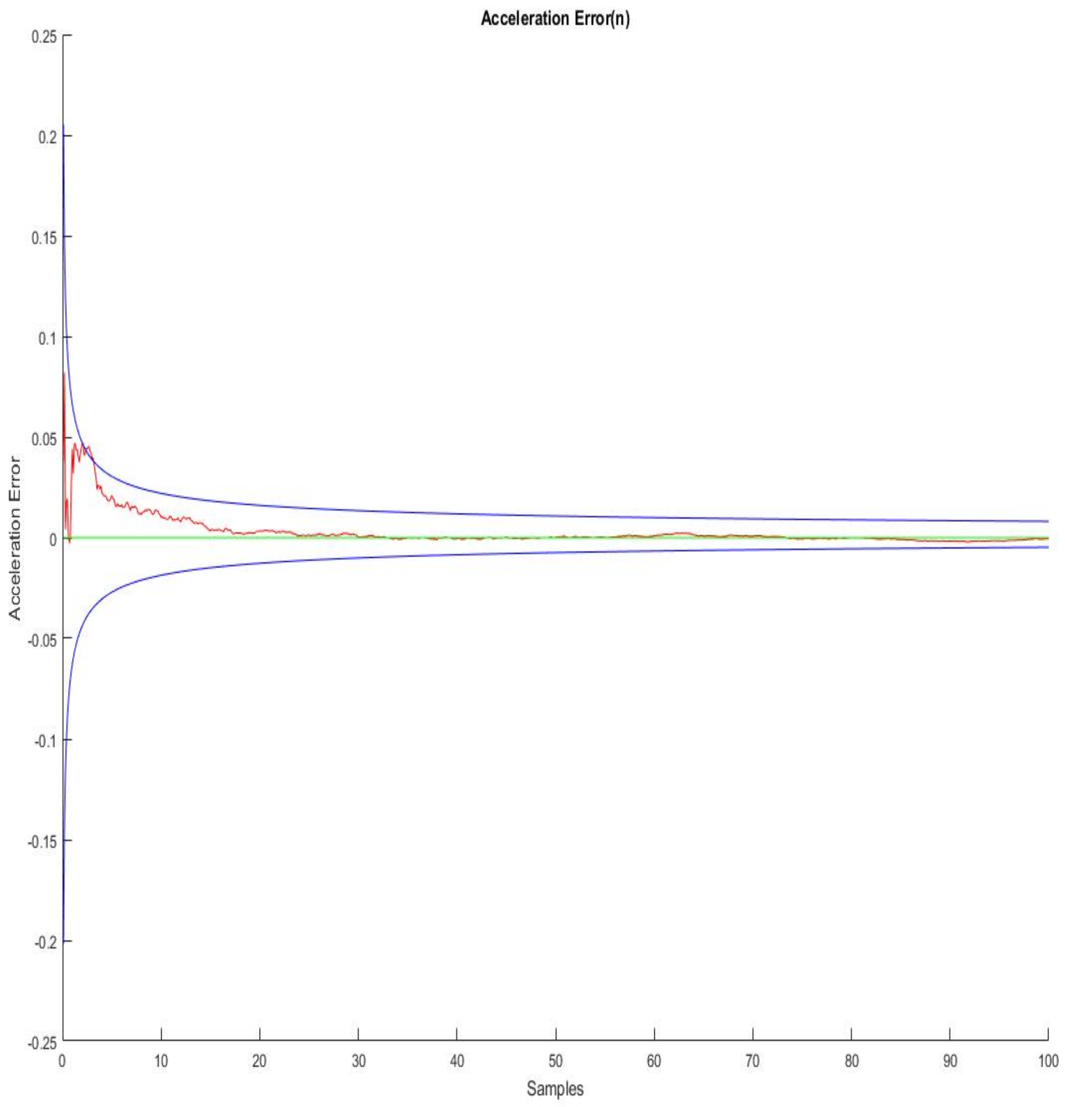

Figure 6: Acceleration Noise Density

2.4 Communication System

2.4.1 XBee Transceiver

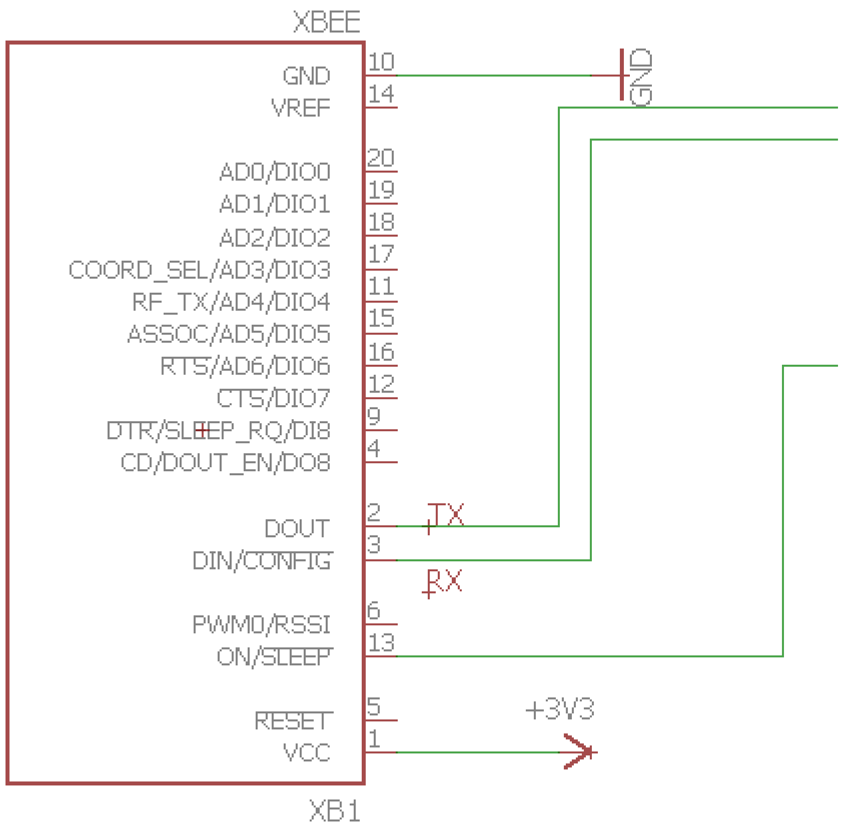

Figure 7: XBee Transceiver

The transceiver that was chosen for the system was the XBee 802.15.4 S2C, this is because the S2C supports multipoint connection and a network level master-slave system. One of the XBee modules can be set to the coordinator, while the rest of the XBee modules can be set to endpoints. This allows for the operation as defined per the original system requirements. The central node can tell the wheel devices to collect data and come out of the idle state. Although Bluetooth supports multipoint connections, it is a more complicated solution and is not defined at the physical layer of the system, rather at the software level. 802.15.4 is the basis for ZigBee which a low powered personal network communication protocol. The XBee remotes can also have their range configured as a function of power and can transmit at as low as -100 dB.

2.5 PCB Design

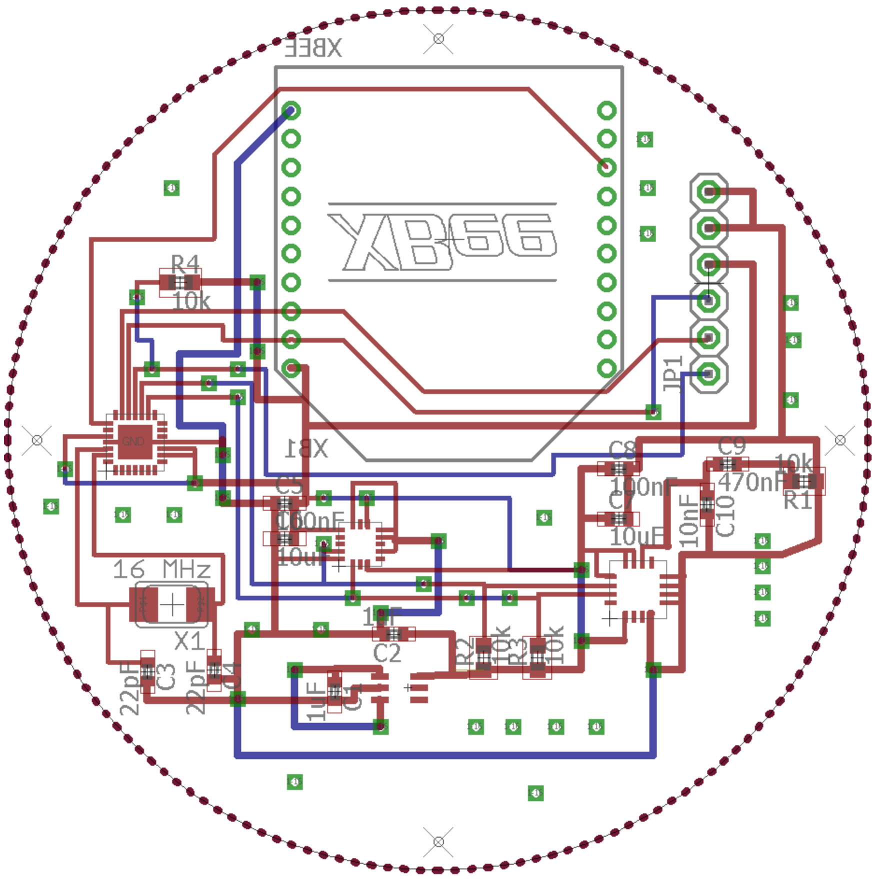

Figure 8: PCB Layout

The PCB that was designed had an area and shape requirement placed on it. It was necessary for the PCB to be placed inside of the wheel logo area on the car, therefore the diameter must be smaller than that of the wheel cap and the board must be circular to optimize the area that is used. A two-layer board with two ground planes was designed to complete the design, all components except for the XBee remote were placed on the top layer to ensure minimal size.

2.6 Software

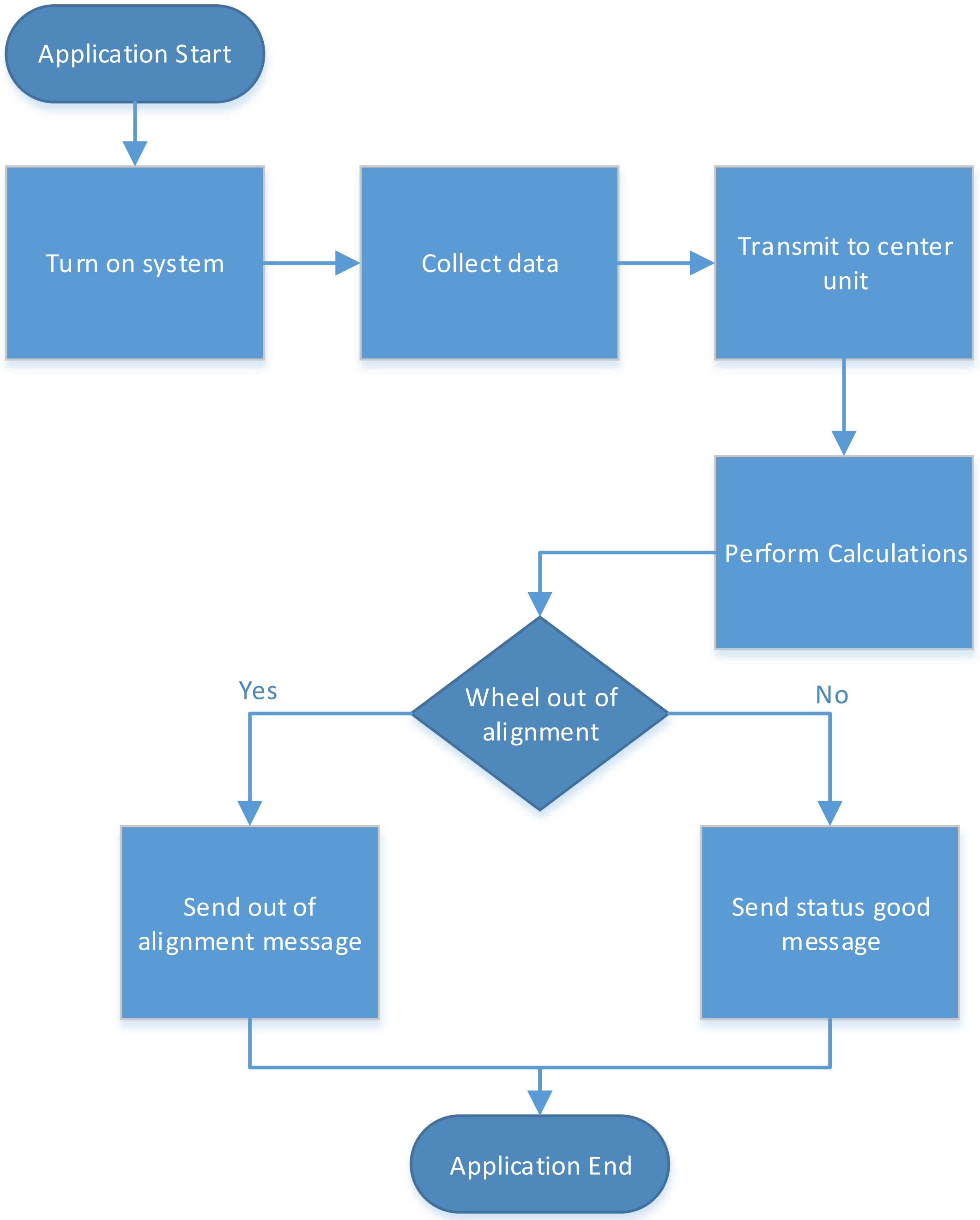

Figure 9: Software Flowchart

The software of the device needed to be a real-time system, such that every task was an aperiodic scheduled task with priority. Taking the measurements had the highest priority, however a command from the central node to stop measuring could act as a SIGKILL which would override the task. However, in general the system has linear behavior that can act as a periodic system, albeit the system would not optimize CPU time unless the calculations could schedule in between the sampling periods.

2.7 Case

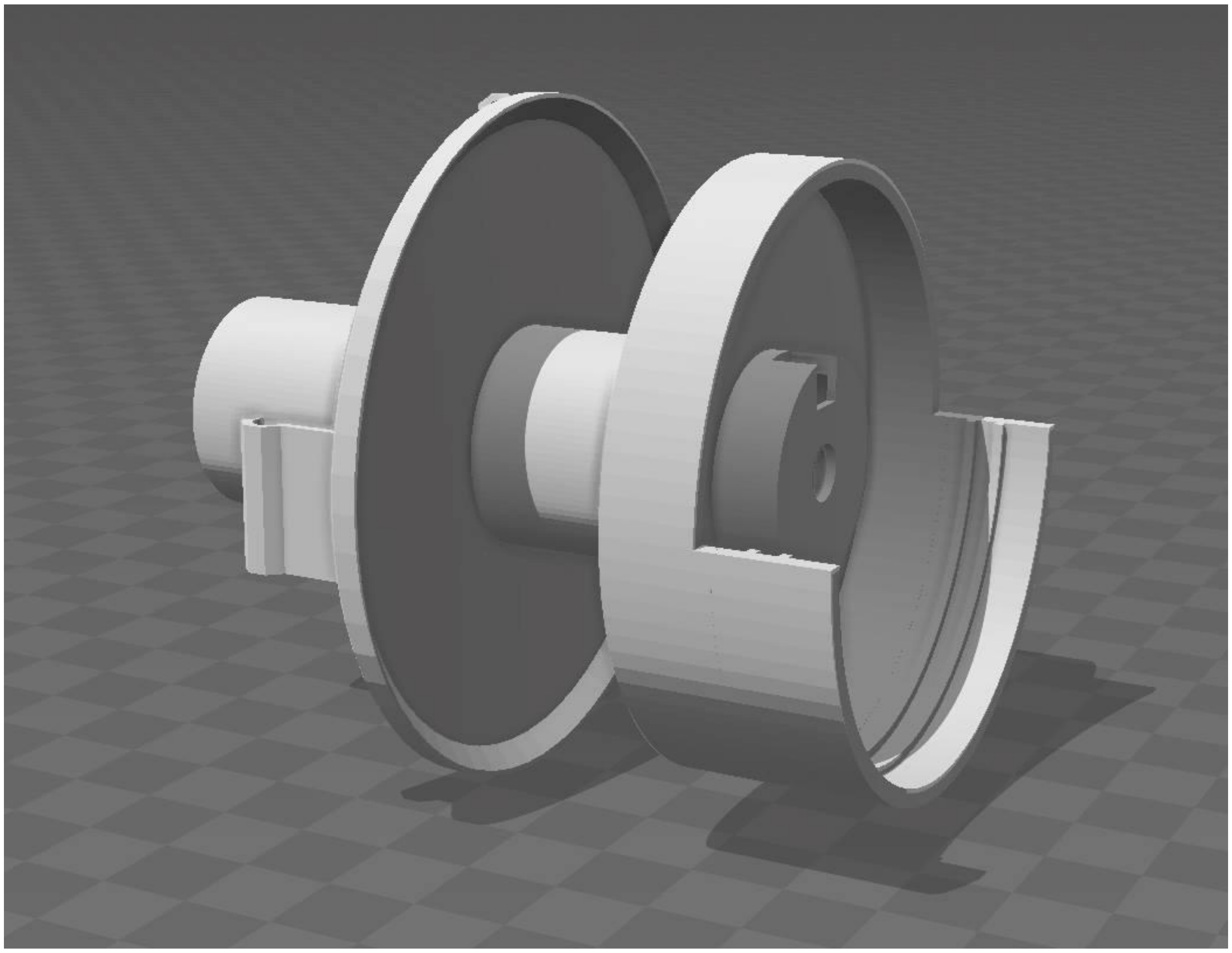

Figure 10: 3D Model of Case

To achieve robustness and stability in our design we modeled the manufacturer wheel cap and designed our cap sensor case around that model. We designed the case to be the same diameter and have the exact same clips so that it will snap into wheel cap holes. In addition, it ensures that the sensors will be aligned to the precision of the machining of the case. Therefore, any type of mounting error can be minimized by using machine precision to achieve a minimum error.

3. Design Verification

The physical verification for the project is complex, because aspects of the project failed to come to fruition. The sensors onboard the nodes failed to collect any data, which prevents any major analysis of the system.

3.1 Power System

The power system's verification procedure was simple, we connected a test probe to the input voltage and a test probe to the output voltage to measure if the voltage was within 5% of 3.7 V and 3.3 V respectively. The voltage that was recorded was 3.64 V and 3.34 V, which are within the allotted margins.

3.2 Communication System

The communication verification procedure was to compare the messages sent versus received between two transceivers placed at least 10 ft. apart. The transceivers sent long strings back and forth to test if there was any drop in communication between the nodes. The messages that were sent matched and the requirement passed its verification test. The hamming distance between the messages was zero indicating that the messages were identical.

3.3 Control System

The microcontroller verification procedure was to test if it could communicate with the other modules and perform the calculations that were necessary to perform the basis of the project. By communicating with the modules, the microcontroller should be able to get the correct identifier response from the sensors and can send/receive data over the communication system and read data from the sensors. The microcontroller that was used did not have bootloader support for the compiler that was chosen. This was an odd occurrence because of the four packages for this microcontroller the other three are supported.

3.4 Sensors

The sensor unit should be able to measure data from the accelerometer and use that data to determine the camber and caster. The camber and caster should be able to be determined from the following equations using the acceleration vectors.

Camber, ϕ

Eq1. $\phi = \tan^{-1}\left(\frac{a_z}{a_{xy}}\right)$

Caster, θ

Eq2. $\theta = \tan^{-1}\left(\frac{\pm \mathrm{a_y}}{\sqrt{a_z^2+a_x^2}}\right)$

These equations can be used to generate the angle values of those measurements. For verification, the values would be compared against an alignment rack value to determine whether the values are within 0.1°. The issue that was encountered that these sensors were very sensitive to external stimuli when unprotected. We were originally able to get output from the sensors, however the nodes encountered some sort of stimuli that damaged them. Thus we were not able to determine whether the sensors could provide data within the 0.1° margin.

4. Costs

It is estimated that the labor cost for development will be approximately $84,00 and the cost per node will be $29.6 for a bulk order. These values can change depending on if a manufacturer is secured or a car manufacturer does all processing in house.

4.1 Parts

Following is a table containing a summarized version of all parts. The major components are listed; however, the exact capacitor values have been combined into a bulk capacitor group. Some devices bulk price can be dropped if partnered with a manufacturing house.

Table 2

Table 2: Parts Costs

Part	**Manufacturer**	**Retail Cost ($)/node**	**Bulk Purchase Cost ($)**	**Actual Cost ($)**
LP5912	Texas Instruments	1.41	0.98	1.41
ATMEGA328P MLF	Atmel	3.73	1.86	3.73
LIS3DSH	STMicroelectronics	2.41	1.43	2.41
XBee 802.15.4	Digi	17.5	10	17.5
3.7V 18650 Li-Ion Battery	Tenergy	8.00	6	8.00
Various Capacitors	Kemet	4.00	0.50	4.00
10 kOhm Resistors	Kemet	0.08	<0.01	0.08
16 MHz Crystal	NDK	1.2	0.83	1.2
PCB Fabrication	PCBway	12	8	12
Total		**$50.33**	**$29.6**	**$50.33**

4.2 Labor

Our labor costs are calculated to be $35.00/hour, 30 hours/week for 16 weeks. For two individuals, this results in a final labor cost of:

$$2\ people * 35\frac{dollars}{hour} * 30\frac{hours}{week} * 16\ weeks * 2.5 = \$84{,}000$$

5. Conclusion

5.1 Accomplishments

While critical component failures prevented us from being able to collect accelerometer data, we are, nevertheless, proud of the work we accomplished. We identified key characteristics our system should have, chose appropriate parts to meet those goals, designed a custom PCB, and demonstrated that our circuits could carry out some key operations such as transforming the battery voltage down to the required 3.3 V and being able to communicate with other chips up to 10 feet away. We are also proud that we put our best effort and countless hours into completing this design project. Given more time to reconsider our design, we are confident that we could achieve a fully-functional alignment sensing system.

5.2 Uncertainties

Although our soldered components appeared to lay flat on their respective PCBs, any deviation in component placement from lying flat on the PCBs would skew our alignment data from actual values. Likewise, any looseness present in our mounting hardware would also skew alignment data from actual values. To combat these issues, our best bet would be to outsource assembly of our circuits and mounting hardware to professional fabrication companies that have more precise manufacturing equipment than what we had at our disposal. By doing so, our mounting hardware would also be less prone to breaking, since it could be cast in acrylonitrile butadiene styrene plastic, which is much more rigid than the polyamide we used while 3D printing our mounting hardware.

5.3 Ethical considerations

Our design is consistent with the IEEE code of ethics [8]. To ensure the safety, health, and welfare of users and the public, our design does not interfere with the operations of the vehicle on which it is installed, and any information gathered from our system only aids in the maintenance of said vehicle. Our claims and estimates are honest and realistic in that our design uses existing technology with known operating characteristics. In addition, no forgeries in analysis or estimations are given. The technology that we use for this application is used for similar concepts but none directly for this application. Our work was carried out diligently, and we present ourselves in a professional manner, citing others where credit is due.

5.4 Future work

If our design had fully functioned as intended, our design could still be improved. By using more sensitive sensors, we could extend the precision of our system to measure primary angles to hundredths of a degree. Since the 4 wheel sensors will be spun around while the car is in motion, it is possible that we could decrease our battery size - and thus, packaging size - if we included a mechanism to convert kinetic energy to stored electric potential. The central chassis-mounted sensor could also draw power from the car battery instead of a separate battery, eliminating the need to change batteries for this unit.

Using accelerometers alone, toe data could not be found. Future designs could use optical sensing methods to determine the distance between the front and back of tires on corresponding sides of the vehicle, allowing for toe values to be calculated.

References

[1] C. Ofria, “A Short Course on Wheel Alignment,” *Wheel Alignment A Short Course | CarParts.com*. [Online]. Available: http://www.carparts.com/alignment.htm. [Accessed: 06-Dec-2016].

[2] “Wheel Alignment,” *Car Struction*. [Online]. Available: http://www.carstruction.com/wheel-alignment/. [Accessed: 06-Dec-2016].

[3] “What is the average length of a car?,” *Reference*. [Online]. Available: https://www.reference.com/vehicles/average-length-car-2e853812726d079d. [Accessed: 06-Dec-2016].

[4]"MEMS digital output motion sensor: ultra-low-power high-performance three-axis ‘nano’ accelerometer,". [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/datasheet/23/c3/ea/bf/8f/d9/41/df/DM00040962.pdf/files/DM00040962.pdf/jcr:content/translations/en.DM00040962.pdf. Accessed: Dec. 8, 2016.

[5]"ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH,". [Online]. Available: http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_summary.pdf. Accessed: Dec. 8, 2016.

[6]"LP5912 500-mA Low-Noise, Low-IQ LDO,". [Online]. Available: http://www.ti.com/lit/ds/symlink/lp5912.pdf. Accessed: Dec. 8, 2016.

[7]"XBEE Reference Guide,". [Online]. Available: https://www.digi.com/pdf/ds_xbee_zigbee.pdf. Accessed: Dec. 8, 2016.

[8] "IEEE IEEE code of ethics," 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. Accessed: Oct. 14, 2016.

Appendix A Requirement and Verification Table

Requirements	Verification	Points
Power 1. Sensor Unit: Turns on when pinged by microcontroller 2. Sensor Unit: Steps down Li-Poly battery voltage from 3.7V±5% to 3.3V±5%.	1. Verification process for Item 2: a. Connect power. b. Start software. c. Check the contents of the sensor's EEPROM registers to determine the power states. 2. Verification process for Item 3: a. Turn car accessory power on. b. Start software. c. Probe Li-Ion battery leads with a lab multimeter to verify that the battery voltage is 3.7V±5%. d. Probe power supply output voltage leads with a handheld multimeter to verify that the output voltage is 3.3V±5%.	1. 5 2. 10
Microcontroller 1. Executes alignment system software. 2. Properly communicates with all other modules.	1. Verification process for Item 1: a. Turn on devices. b. Start software. c. Check software log files to verify that software runs without *critical* errors. 2. Verification process for Item 2: a. Turn battery power on. b. Start software. c. Check software log files to verify that software runs without *communication* errors.	1. 15 2. 15
Communication Module 1. Central receiving antenna receives sensor data from all transmitting antennas up to 10 feet away.	1. Verification process for Item 1: a. Place sensor transmitting antennas 10 feet away from central receiving antenna. b. Connect power. c. Start software. d. Check software log files to verify that software runs without *data transmission* errors.	1. 10

Sensors 1. Sensors measure camber within ±.1° of actual value.	1. Verification process for Item 1: a. Connect power. b. Start software. c. Note camber values listed in software. d. Measure camber values manually or at an alignment shop. e. Compare software and measured values to verify requirement has been met.	1. 15
2. Sensors measure caster within ±.1° of actual value.	2. Verification process for Item 2: a. Connect power. b. Start software. c. Note caster values listed in software. d. Measure caster values manually or at an alignment shop. e. Compare software and measured values to verify requirement has been met.	2. 15
3. Sensors measure toe within ±.1° of actual value.	3. Verification process for Item 3: a. Connect power. b. Start software. c. Note toe values listed in software. d. Measure toe values manually or at an alignment shop. e. Compare software and measured values to verify requirement has been met.	3. 15

Appendix B Code

```
#include "Wire.h"
#include <math.h>

void writeReg_acc(uint8_t reg, uint8_t value);
uint8_t readReg_acc(uint8_t reg);
void writeReg_gyro(uint8_t reg, uint8_t value);
uint8_t readReg_gyro(uint8_t reg);
void readAccel(int16_t *x, int16_t *y, int16_t *z);
double calcCamber(int16_t x, int16_t y, int16_t z);
double calcCaster(int16_t x, int16_t y, int16_t z);

#define GYRO                             0x68
#define ACC                              0x1D
#define LIS3DSH_OUT_X_L          0x28
#define LIS3DSH_OUT_X_H          0x29
#define LIS3DSH_OUT_Y_L          0x2A
#define LIS3DSH_OUT_Y_H          0x2B
#define LIS3DSH_OUT_Z_L          0x2C
#define LIS3DSH_OUT_Z_H          0x2D

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  writeReg_acc(0x20, 0x17);
}

void  loop()
{
  int16_t x,y,z;
  readAccel(&x,&y,&z);
  Serial.print("Accellerometer X Data: ");
  Serial.print(x);
  Serial.print("    Accellerometer Y Data: ");
  Serial.print(y);
  Serial.print("    Accellerometer Z Data: ");
  Serial.print(z);

  camber = calcCamber(x,y,z);
  caster = calcCaster(x,y,z);
  Serial.print('\n');
  delay(100);
}
```

```
// Writes an accelerometer register
void writeReg_acc(uint8_t reg, uint8_t value)
{
        Wire.beginTransmission(ACC);
        Wire.write(reg);
        Wire.write(value);
        Wire.endTransmission();
}


// Reads an accelerometer register
uint8_t readReg_acc(uint8_t reg)
{
        uint8_t reg_value;

        Wire.beginTransmission(ACC);
        Wire.write(reg);
        Wire.endTransmission();
        Wire.requestFrom(ACC, 1);

        while(!Wire.available());

        reg_value = Wire.read();
        Wire.endTransmission();

        return reg_value;
}

// Reads the 3 accelerometer channels
void readAccel(int16_t *x, int16_t *y, int16_t *z)
{
        uint8_t xL;
        uint8_t xH;
        uint8_t yL;
        uint8_t yH;
        uint8_t zL;
        uint8_t zH;

        if((NULL != x) && (NULL != y) && (NULL != z))
        {
                Wire.beginTransmission(I2CAddr);

                xL = readReg(LIS3DSH_OUT_X_L);
        xH = readReg(LIS3DSH_OUT_X_H);
        yL = readReg(LIS3DSH_OUT_Y_L);
        yH = readReg(LIS3DSH_OUT_Y_H);
        zL = readReg(LIS3DSH_OUT_Z_L);
        zH = readReg(LIS3DSH_OUT_Z_H);

        *x = (int16_t)(xH << 8 | xL);
        *y = (int16_t)(yH << 8 | yL);
        *z = (int16_t)(zH << 8 | zL);
    }
}

// Writes an gyroscope register
void writeReg_gyro(uint8_t reg, uint8_t value)
{
    Wire.beginTransmission(GYRO);
    Wire.write(reg);
    Wire.write(value);
    Wire.endTransmission();
}


// Reads an gyroscope register
uint8_t readReg_gyro(uint8_t reg)
{
    uint8_t value;

    Wire.beginTransmission(GYRO);
    Wire.write(reg);
    Wire.endTransmission();
    Wire.requestFrom(GYRO, 1);

    while(!Wire.available());

    value = Wire.read();
    Wire.endTransmission();

    return value;
}
```