# Crane Positional Sensor

Design Document

Team 20
Stellar Industries - Client
Eli Davidson
Andrew Jacobson
Nikhil Sharma
Wyatt Syhlman
sdmay21-20@iastate.edu
Team Website

Revised: 11/15/2020 V3

# Executive Summary

## Development Standards & Practices Used

IEEE 1554-2005: Recommended Practice for Inertial Sensor Test Equipment, Instrumentation, Data Acquisition, and Analysis

IEEE 528-2019: Standard for Inertial Sensor Terminology

## Summary of Requirements

- Product must be mountable from the outside of the body of the crane
- Product must be more economically efficient than off-the-shelf products
- Product must withstand elements of nature between -40° F to +160°F
- Product must connect to a 24V DC power supply
- Product must communicate with a controller via CAN Bus
- Product must be accurate to an error of ±1°

## Applicable Courses from Iowa State University Curriculum

EE 185

EE 201

EE 230

CPRE 281

CPRE 288

## New Skills/Knowledge acquired that was not taught in courses

Working with a client and how to best communicate with them to meet their provided requirements. The use of CAD to develop containment modules and testing equipment.

# Table of Contents

## List of figures/tables/symbols/definitions

# 1 Introduction

## 1.1 Acknowledgement

We would like to acknowledge Stellar Industries for contributing to this project by providing financial assistance, equipment, and technical advice. This group will provide our team with the assistance required to get the project completed.

## 1.2 Problem and Project Statement

Stellar Industries currently does not have the appropriate sensors to communicate with their crane operators. The main goal is to provide Stellar's crane operators with the proper angle at which the crane is rotating around the truck, while it is in operation, to allow for the safe use of the crane. The reason they need this sensor is that the truck is more stable depending on the angle of the crane while moving a load. Figure 1 shows the Stability Capacity Chart that Stellar currently uses to test each different truck's percentage of rated capacity for these angles. That means that Stellar tests at what angles the truck can hold a maximum load and at what angles the operator has to be aware of, for the truck being able to tip over, if the weight of the object is too much or if the object is too far away from the truck. Stellar is currently using the judgment of their crane operators to determine the angle at which the crane is positioned according to the point of origin. In Figure 1, the point of origin is the 0° mark. The sensor also must be able to operate in different elements of the environment.

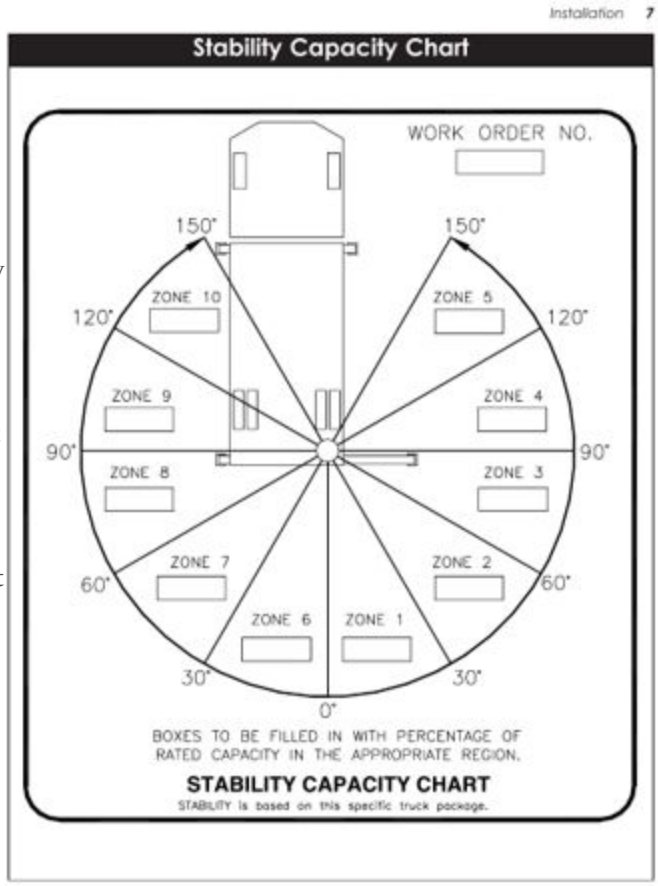



Figure 1: Stability Capacity Chart

Stellar has two additional sensors that could be improved upon. They consist of the angle sensor, the angle at which the boom arm is compared to the box of the truck, and the radial sensor, the distance away from the truck the boom arm is extended. These sensors are a secondary goal, since there is already a method for obtaining this data.

The proposed solution to the problem is to design a sensor that will determine the angle at which the crane rotates and, if possible. implement the secondary sensors as well. The sensor will then be able to provide the crane operator with the exact angle at which the crane is positioned according to the point of origin. To communicate with the crane operator, the sensor has to be able to output to a transceiver and be sent to the controller. We will also build a housing unit that is weather resistant. We hope to build a functional and safe rotational sensor that will set Stellar apart from their competitors. The main focus of this project is the rotational sensor.

## 1.3 Operational Environment

The cranes that Stellar Industries produces are for outdoor use. Therefore, the fabricated sensors need to withstand any and all environmental conditions. This includes extreme heat, freezing cold, rain, fog, sleet, snow, and high winds. These sensors need to be fully operational and will spend almost all, if not all, of their usable life outside. Therefore, they need to be evaluated for such at an early stage.

## 1.4 Requirements

- It must be more cost-efficient than the current sensor setup.
- It must be able to be easily mounted on a truck.
- The sensors are expected to communicate to a handheld interface, which will display the sensors' information and outputs.
- The budget for development and testing must not exceed $2000.
- It must be able to work in -40° F to +160°F
- It must receive its power from a 24V DC line
- It must output via CAN bus, using either J1939 or Open Can

## 1.5 Intended Users and Uses

The intended users for this particular product are the employees of Stellar Industries that are incharge of operating the crane. They will be able to see an accurate reading of the angle which will make their work a lot more efficient and safe. Furthermore, the data that is gathered from the sensor can be used by the Stellar's mechanical department to make sure that their cranes are responding to commands accurately.

## 1.6 Assumptions and Limitations

Assumptions

1. The end product is for Stellar Industries
2. The sensor covers need to withstand the various weather conditions
3. Cost of sensors should be less than off-the-shelf solutions

Limitations

1. The maximum degree of rotation of the **rotational sensor** will be dependent on the truck size
2. The maximum degree of rotation of the **angular sensor** will be from -10° to +80°
3. The maximum length of the **radial sensor** will be dependant on the truck size
4. Able to work in -40° F to +160°F
5. Budget of $200

## 1.7 Expected End Product and Deliverables

There are one mandatory and two optional deliverables for this project:

1. Rotational Sensor (Mandatory)
2. Angular Sensor
3. Radial Sensor

Each of these sensors, in their end product form, should be able to do as stated above in 1.6 Limitations. The sensors should be able to provide feedback to the UI so the operator knows how the crane is operating. These sensors are to be delivered to Stellar Industries in May of 2021.

The rotational sensor should be able to rotate the crane from 0° to 370° when starting at the boom cradle. Once the crane is unstowed and rotated clockwise to the centerline of the truck, the crane should be able to turn a minimum of another 150° clockwise. This will allow the maximum usage of the crane. The crane sensor should be able to read the centerline of the truck as the 0° mark so the operator knows how far each way the crane is still able to move while the crane is in operation.

The angular sensor should be able to rotate from -10° to +80°. When the crane angle is at 0°, the crane should be at the horizontal position and parallel to the truck bed. The angle of the crane will tell the operator how much payload the crane can handle and if the angle is off, the operator could end up seriously hurt.

The radial sensor is truck dependent, so the sensor should be able to read when the crane stops extending and relay the distance back to the operator.

# 2 Project Plan

## 2.1 Task Decomposition

Our project has multiple tasks. Our first task will be evaluating the cost of sensors currently in use in Stellar cranes. Our next task will be to design a sensor that will have a lower manufacturer and operating cost then what is currently being used. The task of designing a sensor can be broken up into multiple subtasks such as evaluating the components used in the sensors currently in use, forming ideas on a new design, creating documentation of our design, creating any in-house components needed for our new design, creating a prototype, and testing.

## 2.2 Risks And Risk Management/Mitigation

For the task of evaluating the cost of sensors currently in use, the probability of risk is 0.1. It seems unlikely that we would have issues understanding what is currently in use. As for forming ideas for a new design, the probability of risk would be ~0.7 because our ideas might not work out as planned. For this, we would need to consider any risk that we may have when coming up with ideas. For creating documentation, there is no risk, as we will just need to make proper documentation. Creating in-house components has a risk of 0.6, as we would have to do our own fabrication and testing to make sure the component works. One alternative would be to use off-the-shelf components for our design. Finally, testing has a risk of 0.5 because our test might fail. If our testing fails, we would have to look at our design and debug it. Another possibility is to create prototypes of multiple designs, as we may have more than one idea.

## 2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

For our first task, our milestone is evaluating the cost of the current sensors in use at Stellar. This milestone will be accomplished by doing research on the current sensors in use and coming up with a total cost estimate on the sensor system as a whole. Our next milestone will be forming ideas and creating design documents for our ideas. This will include the documentation for

our components, a detailed description of how they work together, and instructions on how to create our design. This can be measured in the percentage completion of our design documents. For creating our design/prototype, milestones can be measured by the accuracy of our design such as 99% accuracy of the rotational sensor and sending the user data every 10 milliseconds. Finally, for testing, milestones can be measured by the accuracy of the sensors used and the speed at which they are able to take measurements as stated in our design and prototyping milestones.

### 2.4 Project Timeline/Schedule

Our timeline for our project is from August 2020 to May 2021. In that time we expect to have defined the problem, ideated upon a solution, designed potential solutions, tested potential solutions, chosen the best solution, and built a final solution. Our timeline is laid out below in Figure 2, note, it is subject to change. We expect to have the deliverable ready around March 15th, 2021.

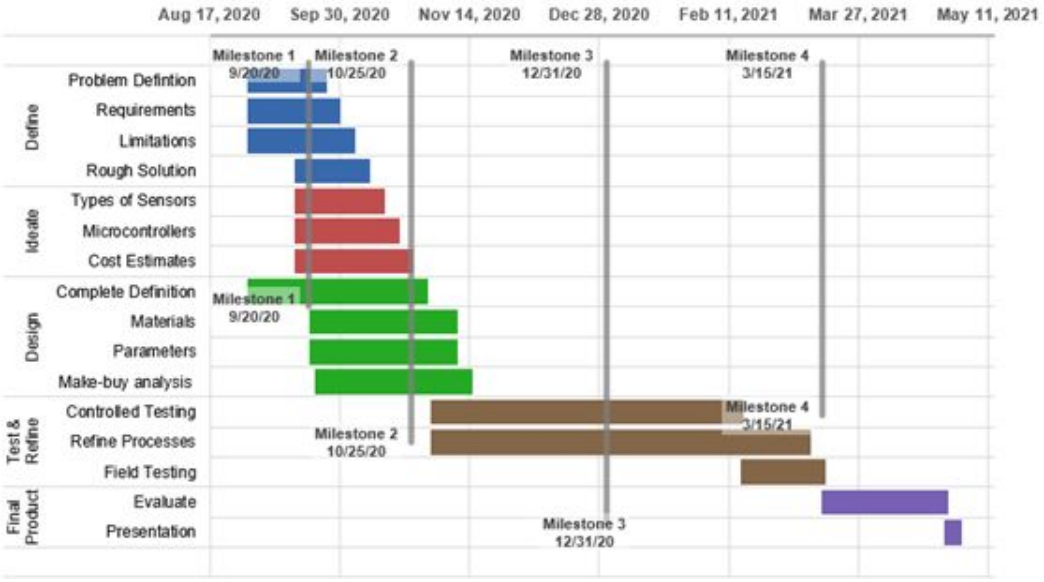



Figure 2: Timeline

| CATEGORY | TASK | START | END |
|---|---|---|---|
| **Define** | Problem Defintion | 8/30/20 | 9/25/20 |
| | Requirements | 8/30/20 | 9/30/20 |
| | Limitations | 8/30/20 | 10/5/20 |
| | Rough Solution | 9/15/20 | 10/10/20 |
| **Ideate** | Types of Sensors | 9/15/20 | 10/15/20 |
| | Microcontrollers | 9/15/20 | 10/20/20 |
| | Cost Estimates | 9/15/20 | 10/25/20 |
| **Design** | Complete Definition | 8/30/20 | 10/30/20 |
| | Materials | 9/20/20 | 11/9/20 |
| | Parameters | 9/20/20 | 11/9/20 |
| | Make-buy analysis | 9/22/20 | 11/14/20 |
| **Test & Refine** | Controlled Testing | 11/1/20 | 2/15/21 |
| | Refine Processes | 11/1/20 | 3/10/21 |
| | Field Testing | 2/15/21 | 3/15/21 |
| **Final Product** | Evaluate | 3/15/21 | 4/26/21 |
| | Presentation | 4/26/21 | 5/1/21 |

Table 1: Task Dates

## 2.5 Project Tracking Procedures

We are currently using a Discord server and Google Drive that will help us develop effective communications and a tracking methodology and share any files such as our design document and code we are developing. We have also been holding meetings twice a week on Tuesday and Sunday where we start by going through what needs to be accomplished throughout the meeting and the week to stay on schedule with our project.

## 2.6 Personnel Effort Requirements

| **Task** | **Man Hours** |
|---|---|
| Research Requirements and Limitation | 10 |
| Research Sensor Types | 25 |
| Research Microcontrollers | 25 |
| Create Documentation | 20 |
| Create Design | 50 |
| Implement Design | 30 |
| Evaluate Test Information | 20 |
| Build Prototype | 10 |

Table 2: Personal Effort Requirements

Researching requirements will not take us much time as we will just be understanding the requirements given to us and possible solutions. Researching sensor types and microcontrollers will take more man hours as we will be researching to understand different options that we could be implanting into our design. Documenting our research will not take as much time as researching but is important to understanding our findings. The bulk of the project will be creating our design as this will require us to come up with ideas and figure out ways to implement them into a cohesive design. Implementing and testing our design will not take as much time as creating our design but is just as important because using our test data, we will be able to understand if our design will meet the requirements. Finally, we determined that building our prototype will not take much time as we will just be implementing our design into a more formal package.

## 2.7 Other Resource Requirements

When we look at resource requirements, we are looking at the requirements such as working stations, and equipment at Iowa State that could benefit us. We can use the workstations at the TLA where we will be able to use soldering stations and yet also be able to code.

## 2.8 Financial Requirements

We have a few main components which we deem essential in order to complete the project:

1. Arduino Uno Rev3 SMD: $17.52
2. IME18-08NPSZC0S: $53.90
3. YF2A14-020VB3XLEAX: $20.12
4. MPU-6050 - Accelerometer, Gyroscope, 3 Axis Sensor Evaluation Board: $10.00
5. PDS1-S24-S5-S: $4.31

These component costs have reached $105.85.

# 3 Design

## 3.1 Previous Work And Literature

Our research has consisted on understanding what type of sensor would best suit our needs. We have settled on two different types of sensors. The gyroscopic sensor and the inductive sensor. Stellar wanted us to develop a different type of sensor that was not a gear driven sensor or that could be found on their competitors cranes. The gyroscope sensor has been used in many things including, but not limited to, phones, aircrafts and ships. The inductive sensor is used in assembly lines and other automotive applications. Both of these sensors have not been used in the way we are trying to implement them. Trying to find previous works or products in the market proved to be difficult and inconclusive. However, there were some studies done on the gyroscope that helped us improve upon our design.

We had to do some research on how to calculate the angle of the gyroscope sensor. There were some documents that proved to be helpful from the IEEE website in determining the best method for calculating the angle. It was determined that the angular position of the gyroscope could be found by integrating the angular velocity.

We have yet to find a way to implement the inductive sensor in a way that will not require Stellar Industries to take apart their crane. They want to avoid this so we are still trying to find ways to implement this type of sensor without doing so. For that reason we do not have any design plan for this sensor as of yet but will continue researching ideas as an alternative to the gyroscopic sensor.

## 3.2 Design Thinking

During the “define” phase we came up with a couple of different parameters that we would have to abide by. We have to be able to use this sensor to compute the correct angle at which the crane is positioned when the operator is rotating the crane. The sensor also needs to be able to operate with some vibration from the crane moving. The truck will not be running so there will be no extra vibration from that. We also have to be able to place this sensor on the crane without altering any parts of the crane in a substantial way. The sensor has to be a type of “plug and go” solution so the company doesn’t have to spend more money on rebuilding/redesigning their cranes.

During our “ideate” phase we came up with many ideas that consisted of a gear driven sensor but after talking to Stellar during our site visit, we found out that they would like to come up with a different type of sensor than what is already being used and can be unique to Stellar Industries. This caused us to rethink what we did in the “define” phase but not too much.

## 3.3 Proposed Design

Our main idea for this design is using a gyroscopic sensor. The gyroscope sensor satisfies our needs because it allows us to convert angular velocity into degrees moved. This option also might be able to help us achieve another goal. A gyroscope does not work on only one axis and that will allow us to possibly use it for the angular sensor as well as the rotational. We are planning on trying to implement this onto the joint of the crane where the boom arm meets the base using a control box to protect the gyroscope from weather, dust and other damaging elements.

To implement our design, we are using a microcontroller to run code to utilize the gyroscope and to convert the raw data from the gyroscope to the cranes current angle. We are currently using an Arduino as our microcontroller. Arduinos run on 5V DC. To get the desired input voltage from the 24V DC input, we are using a buck converter to step down 24V DC to 5V DC. As for the output, we need to output our data using CAN Bus protocol J1939. So far we have implemented both the microcontroller and the gyroscope to test our design and to see if it is even feasible. Section 4 will go into detail about testing and our results. We plan on using a CAN Bus Arduino shield that will allow us to implement CAN Bus as our output. Below is figure 3, which is a system block diagram of our current design.

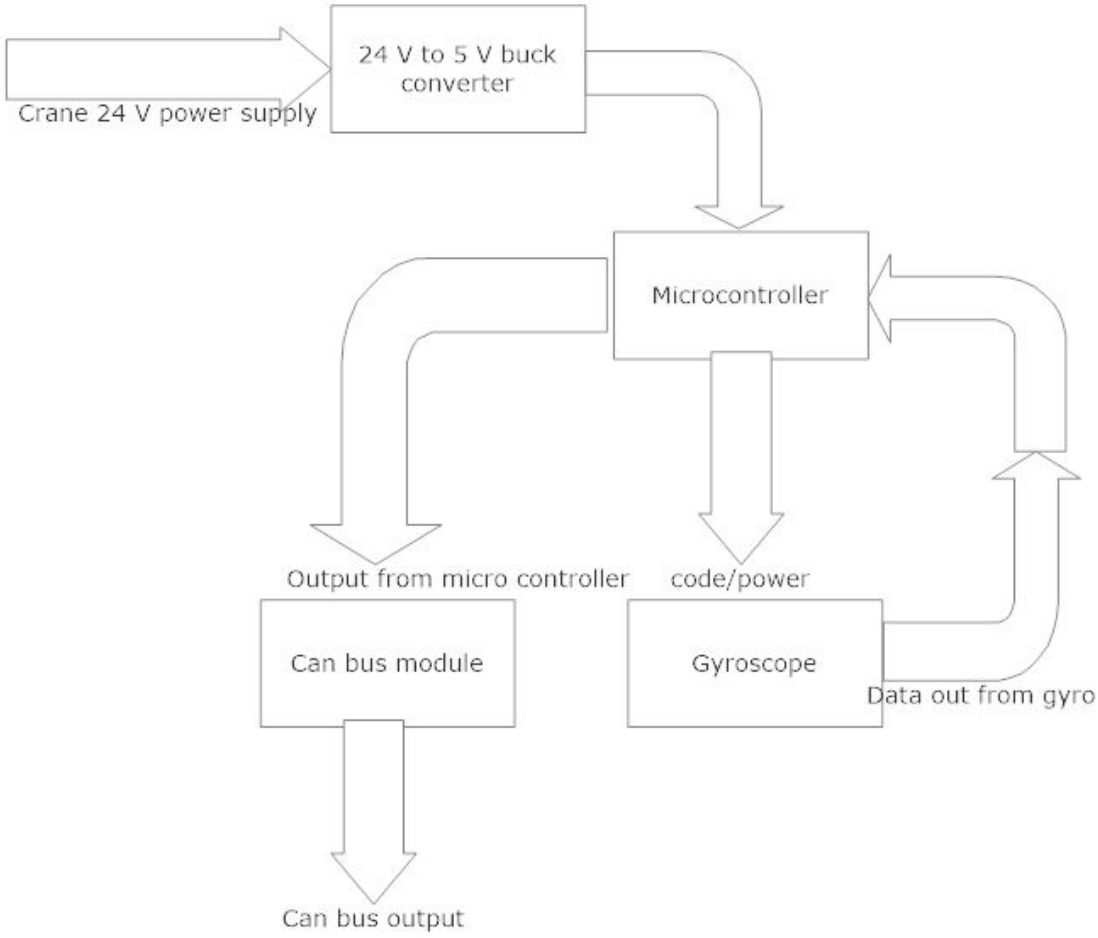



Figure 3: System Block Diagram

Based on the requirements from section 1.4, our design will meet the requirements stated. One requirement that has not been mentioned in this section is how the sensor will be mounted. We plan on using 3M double sided heavy duty waterproof mounting tape. Using that solution, users will be able to easily mount our control box to the boom arm of the crane.

As stated in the Executive Summary, we view two IEEE standards as relevant to our project. The first is a recommended practice on how to test inertial sensors, and how to analyze the data received. While this standard does touch on many other kinds of sensors besides gyroscopic, there is still a sizable section on gyroscopic sensors and how to perform tests on them. The second is a glossary on terms used for inertial sensors.

## 3.4 Technology Considerations

When considering technology's role in our design process, we went back to the design requirements stressed to us in the prompt from Stellar Industries. Their most important need was that of a rotational sensor and for it not to interfere with the physical workings of the crane. In other words, they wanted us to use components that are independent to the working of the crane. Taking this into consideration, we went ahead and decided on potentially using a gyroscope component that would attach to the outside of the crane. While what is stated above is considered a strength of using a gyroscope sensor, it does come with a weakness; the gyroscope will be exposed to the elements of nature. As stated in section 3.3, our solution is to use a waterproof control box to keep any moisture or debris away from the physical circuitry.

## 3.5 Design Analysis

Based on our testing from section 4, we have concluded that our design will work. We were successfully able to gather the current angle of the gyroscopic sensor. We have yet to implement the 24V to 5V buck converter and the Can Bus module, but we have proved the core functionality of using a gyroscope as a rotational sensor. A modification we would like to implement would be a different microcontroller and gyroscope, as we have been using an Arduino for mostly testing purposes. An Arduino is most often used as a hobbyist microcontroller and we would like to implement a cheaper alternative or possibly a custom microcontroller specific to our project. Also, we would potentially like to use a different gyroscope, since the one we are currently using is six-axis and we would only need a two-axis for a rotational sensor. Also of note, if we keep our current gyroscope, then we could possibly implementan angular sensor. If we decide to go with a two axis gyroscope, we would not be able to implement that functionality.

## 3.6 Development Process

We have not really stuck to any development process rationale, since many of them rely on in person interaction. However, we have chosen to use agile as a framework for understanding how to design and the follow the key beliefs that accompany that philosophy.

## 3.7 Design Plan

The use-cases in our design come from the user operating the crane. When the user operates the crane, they can either move the crane left or right, both of which will be use-cases. Based on if the crane is moved left or right, the rotational angle would either increase or decrease. based on figure 3, we plan to implement the use of a microcontroller to process the data coming from the gyroscopic sensor, the microcontroller will then output the processed data through CAN

Bus to Stellar's transmitter for the user to then see the rotational angle of the crane on the crane controller.

# 4 Testing

## 4.1 Unit Testing

In regards to unit testing, the main component that needs to be tested is the gyroscopic sensor itself, and determining how to measure an angle from the data received. In order to test the functionality of this part, we ran a program that would take the sensor data and integrate it. Since the output of the gyro sensor is linearly related to the velocity of the movement, the integral should correspond to the position of the sensor. We then placed the sensor on a rotating arm above a protractor, and rotated the angle back and forth at different speeds.

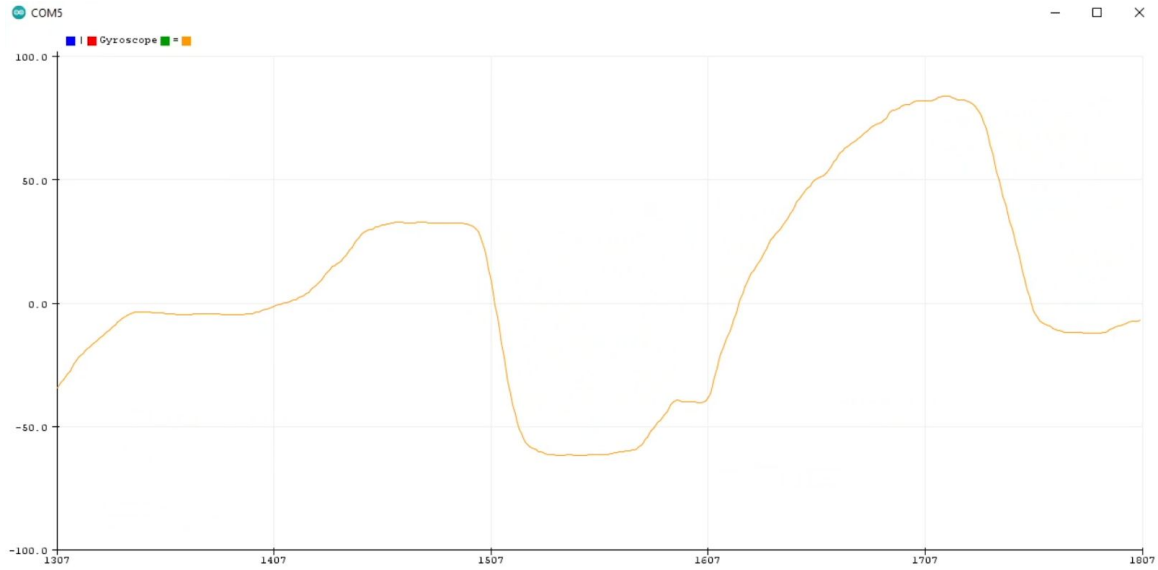



Figure 4: Sample Data from Test Program

While not a rigorous test, this does show that a gyroscopic sensor can be used fairly reliably to output the correct data that we need. More rigorous testing will be performed once parts have been finalized.

## 4.2 Interface Testing

Our design will need to output through a CAN Bus. There are many components that will convert Arduino data to CAN Bus, if we continue to use Arduino. If that is not the case, more testing will need to be done to ensure that our processor can communicate via CAN Bus.

## 4.3 Acceptance Testing

We will attempt to simulate, as best as possible, outside conditions for this project. We will also travel to Stellar Industries again in the future to mount our prototype and make sure that our solution will suit their needs the best.

### 4.4 Results

Overall, the test results have shown that our solution is viable and able to effectively track the rotational position of the crane. As mentioned above, more testing will be done within the coming months to ensure that this gyro sensor will meet ALL of our project requirements, namely communication with CAN Bus and reliable data after long periods of use. That being said, all rotation tests performed so far seem to indicate this to be an effective solution.

Important considerations that will need to be tested in regards to unit testing on the gyroscope sensor will be whether or not this solution can withstand vibration of the boom arm and if the truck is level during operation.

## 5 Implementation

For next semester, we plan to initially integrate in the 24V to 5V buck converter and the CAN Bus protocol J1939 output. Then we will focus on a housing unit for the entire system. We will also do more testing when we implement the buck converter and CAN Bus. One test we will need to do is testing on the crane itself, this will require us to travel to Stellar once again to perform these tests. If we have time at the end of our project, we plan to work on implementing either an angular or positional sensor into our design.

## 6 Closing Material

### 6.1 Conclusion

Our goals for this project were to create a rotational sensor for Stellar Industries that could be mounted on the outside of their cranes. It additionally needed to be less expensive to implement than off-the-shelf solutions, while still being accurate to the degree. We determined, mainly through the first criteria listed, that the best external sensor would be that of a gyroscope. The accuracy of a gyroscope in short-term use, coupled with their lower relative cost, is why we decided that a gyroscope sensor would fit the best. Through testing, we confirmed that a gyroscopic sensor would provide us with rotational information, and potentially provide additional sensor information later down the road.

### 6.2 References

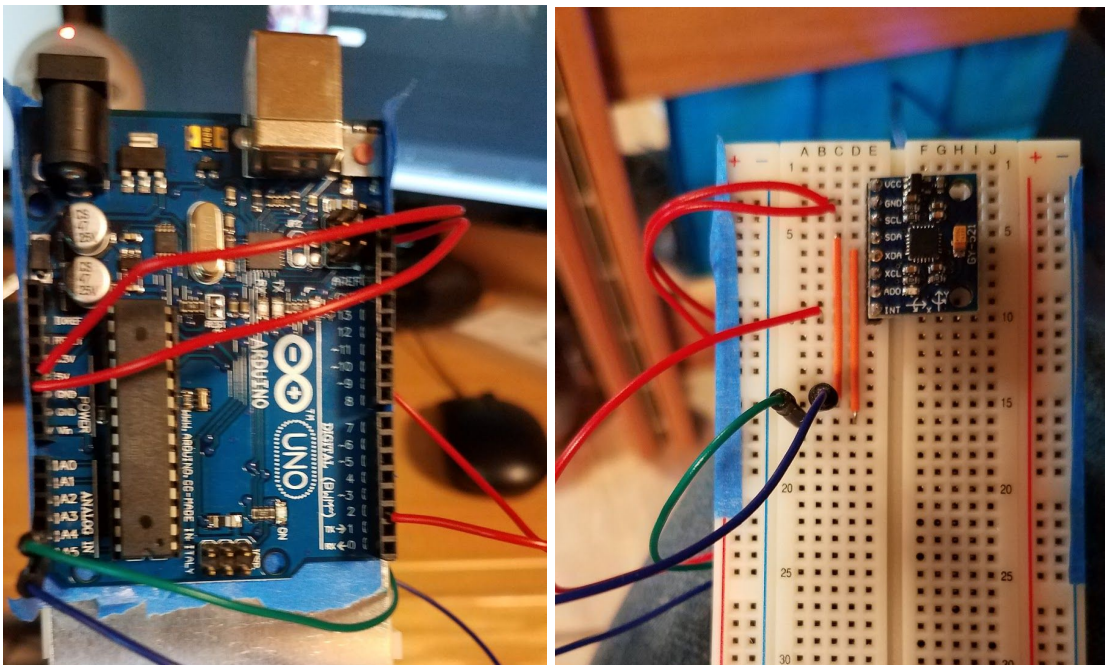
S. Mischie, "On using a gyroscope to measure the angular position," *2012 10th International Symposium on Electronics and Telecommunications*, Timisoara, 2012, pp. 61-64, doi: 10.1109/ISETC.2012.6408139.

N. Ismail, A. Nurhakim and H. M. Saputra, "The Calculation of Gyroscope Sensor Angles Using Several Integral Methods," *2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, Yogyakarta, Indonesia, 2018, pp. 1-5, doi: 10.1109/TSSA.2018.8708754.

"Arduino Spirit Level Using Gyroscope with LED." *Arduino Spirit Level Using Gyroscope with LED ~*, mousa-simple-projects.blogspot.com/2018/07/arduino-spirit-level-using-gyroscope.html

Jones, Phillip. “Living with Noise.” Senior Design. October 2020, seniord.ee.iastate.edu/resources/Jon17A.pdf.

## 6.3 Appendices



Front Back

Figure 5: Gyroscope Testing Design Layout

### Gyroscope Testing Design Code

```
#include <Wire.h>
//Declaring some global variables
int gyro_x, gyro_y, gyro_z;
long gyro_x_cal, gyro_y_cal, gyro_z_cal;
boolean set_gyro_angles;

long acc_x, acc_y, acc_z, acc_total_vector;
float angle_roll_acc, angle_pitch_acc;

float angle_pitch, angle_roll;
int angle_pitch_buffer, angle_roll_buffer;
float angle_pitch_output, angle_roll_output;

long loop_timer;
int temp;
double angle_five = 0, angle_six;

void setup() {
  Wire.begin();                                        //Start I2C as master
  setup_mpu_6050_registers();                          //Setup the registers of the MPU-6050
  for (int cal_int = 0; cal_int < 1000 ; cal_int ++){  //Read the raw acc and gyro data from the MPU-6050 for 1000
times
    read_mpu_6050_data();
    gyro_x_cal += gyro_x;                              //Add the gyro x offset to the gyro_x_cal variable
    gyro_y_cal += gyro_y;                              //Add the gyro y offset to the gyro_y_cal variable
    gyro_z_cal += gyro_z;                              //Add the gyro z offset to the gyro_z_cal variable
    delay(3);                                        //Delay 3us to have 250Hz for-loop
```

```
  }

  // divide by 1000 to get average offset
  gyro_x_cal /= 1000;
  gyro_y_cal /= 1000;
  gyro_z_cal /= 1000;
  Serial.begin(9600);
  loop_timer = micros();                                    //Reset the loop timer
}
void loop(){
  read_mpu_6050_data();
 //Subtract the offset values from the raw gyro values
  gyro_x -= gyro_x_cal;
  gyro_y -= gyro_y_cal;
  gyro_z -= gyro_z_cal;

  //Gyro angle calculations . Note 0.0000611 = 1 / (250Hz x 65.5)
  angle_pitch += gyro_x * 0.0000611;                                  //Calculate the traveled pitch angle and add this to the angle_pitch
variable
  angle_roll += gyro_y * 0.0000611;                                   //Calculate the traveled roll angle and add this to the angle_roll
variable
  //0.000001066 = 0.0000611 * (3.142(PI) / 180degr) The Arduino sin function is in radians
  angle_pitch += angle_roll * sin(gyro_z * 0.000001066);               //If the IMU has yawed transfer the roll angle to the pitch
angle
  angle_roll -= angle_pitch * sin(gyro_z * 0.000001066);               //If the IMU has yawed transfer the pitch angle to the roll
angle
  //Accelerometer angle calculations
  acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z));  //Calculate the total accelerometer vector
  //57.296 = 1 / (3.142 / 180) The Arduino asin function is in radians
  angle_pitch_acc = asin((float)acc_y/acc_total_vector)* 57.296;       //Calculate the pitch angle
  angle_roll_acc = asin((float)acc_x/acc_total_vector)* -57.296;       //Calculate the roll angle
    angle_pitch_acc -= 0.0;                                            //Accelerometer calibration value for pitch
  angle_roll_acc -= 0.0;                                              //Accelerometer calibration value for roll
  if(set_gyro_angles){                                                //If the IMU is already started
    angle_pitch = angle_pitch * 0.9996 + angle_pitch_acc * 0.0004;     //Correct the drift of the gyro pitch angle with the
accelerometer pitch angle
    angle_roll = angle_roll * 0.9996 + angle_roll_acc * 0.0004;        //Correct the drift of the gyro roll angle with the
accelerometer roll angle
  }
  else{                                                               //At first start
    angle_pitch = angle_pitch_acc;                                     //Set the gyro pitch angle equal to the accelerometer pitch angle
    angle_roll = angle_roll_acc;                                       //Set the gyro roll angle equal to the accelerometer roll angle
    set_gyro_angles = true;                                            //Set the IMU started flag
  }
  //To dampen the pitch and roll angles a complementary filter is used
  angle_pitch_output = angle_pitch_output * 0.9 + angle_pitch * 0.1;   //Take 90% of the output pitch value and add 10% of
the raw pitch value
  angle_roll_output = angle_roll_output * 0.9 + angle_roll * 0.1;      //Take 90% of the output roll value and add 10% of the
raw roll value
  //Serial.print(" | Angle  = "); Serial.println(angle_pitch_output);
  //Serial.print(" | Accelerometer  = "); Serial.println(gyro_x);
  if ((gyro_x <= 20) && (gyro_x >= -20)) {
    gyro_x = 0;
  }
  angle_five += gyro_x;                                               //Creates an integrator

  angle_six = angle_five * 90 * 90 / (200000 * 97);
  Serial.print(" | Gyroscope = "); Serial.println(angle_six);

 while(micros() - loop_timer < 4000);                                 //Wait until the loop_timer reaches 4000us (250Hz) before starting
the next loop
 loop_timer = micros();//Reset the loop timer

}
```

```
void setup_mpu_6050_registers(){
  //Activate the MPU-6050
  Wire.beginTransmission(0x68);                                        //Start communicating with the MPU-6050
  Wire.write(0x6B);                                                    //Send the requested starting register
  Wire.write(0x00);                                                    //Set the requested starting register
  Wire.endTransmission();
  //Configure the accelerometer (+/-8g)
  Wire.beginTransmission(0x68);                                        //Start communicating with the MPU-6050
  Wire.write(0x1C);                                                    //Send the requested starting register
  Wire.write(0x10);                                                    //Set the requested starting register
  Wire.endTransmission();
  //Configure the gyro (500dps full scale)
  Wire.beginTransmission(0x68);                                        //Start communicating with the MPU-6050
  Wire.write(0x1B);                                                    //Send the requested starting register
  Wire.write(0x08);                                                    //Set the requested starting register
  Wire.endTransmission();
}


void read_mpu_6050_data(){                                             //Subroutine for reading the raw gyro and accelerometer data
  Wire.beginTransmission(0x68);                                        //Start communicating with the MPU-6050
  Wire.write(0x3B);                                                    //Send the requested starting register
  Wire.endTransmission();                                              //End the transmission
  Wire.requestFrom(0x68,14);                                           //Request 14 bytes from the MPU-6050
  while(Wire.available() < 14);                                        //Wait until all the bytes are received
  acc_x = Wire.read()<<8|Wire.read();
  acc_y = Wire.read()<<8|Wire.read();
  acc_z = Wire.read()<<8|Wire.read();
  temp = Wire.read()<<8|Wire.read();
  gyro_x = Wire.read()<<8|Wire.read();
  gyro_y = Wire.read()<<8|Wire.read();
  gyro_z = Wire.read()<<8|Wire.read();
}
```

**Serial Monitor Output**

```
| Gyroscope = 78.01
| Gyroscope = 77.49
| Gyroscope = 76.83
| Gyroscope = 76.01
| Gyroscope = 75.02
| Gyroscope = 73.86
| Gyroscope = 72.58
| Gyroscope = 71.28
| Gyroscope = 69.76
| Gyroscope = 68.08
| Gyroscope = 66.40
| Gyroscope = 64.78
| Gyroscope = 62.90
| Gyroscope = 60.95
| Gyroscope = 59.07
| Gyroscope = 57.39
| Gyroscope = 55.82
| Gyroscope = 54.18
| Gyroscope = 52.41
| Gyroscope = 51.03
| Gyroscope = 49.97
| Gyroscope = 49.20
| Gyroscope = 48.62
| Gyroscope = 48.19
| Gyroscope = 47.79
| Gyroscope = 47.48
| Gyroscope = 47.15
| Gyroscope = 46.69
| Gyroscope = 46.01
| Gyroscope = 44.99
| Gyroscope = 43.79
| Gyroscope = 42.32
| Gyroscope = 40.62
| Gyroscope = 39.05
| Gyroscope = 37.87
| Gyroscope = 37.05
| Gyroscope = 36.77
| Gyroscope = 36.64
| Gyroscope = 36.42
| Gyroscope = 36.06
| Gyroscope = 35.85
| Gyroscope = 35.42
| Gyroscope = 34.94
| Gyroscope = 34.56
| Gyroscope = 34.16
| Gyroscope = 34.37
| Gyroscope = 33.90
| Gyroscope = 33.08
```