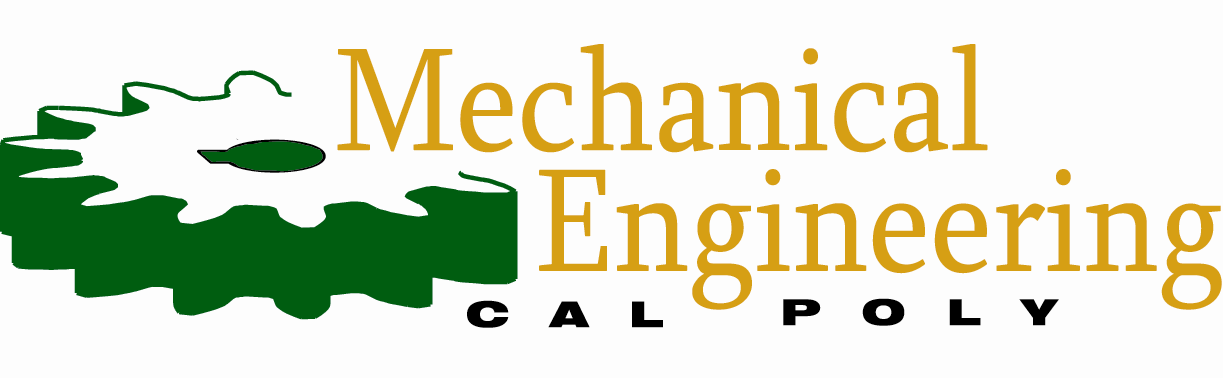
**MEMORANDUM**

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
| **To:** | Kennedy, Jane E. | |
|  | jekenned@calpoly.edu | |
| **From:** | Edward Ruano | Robert Trujillo |
|  | eruano@calpoly.edu | rotrujil@calpoly.edu |
|  | Drew Isaak | Anthony Lombardi |
|  | aisaak@calpoly.edu | atlombar@calpoly.edu |
| **Date:** | May 20th, 2016 | |
| **RE:** | **Lab 5: Using a Sensor** | |
|  |  | |

In this lab, we were tasked with interfacing an external sensor to our project. The sensor we choose to implement was an inertial measurement unit which would provide us with information that would make guiding our vehicle much easier. The IMU we chose to implement was the Adafruit BNO055 based on the Bosch Sensortec chip of the same name.

Creating the driver for this chip was not very difficult as there are many libraries already written for microcontrollers such as the Arduino. Although the code in these libraries will not work on the Atmega1281 chip we have on the custom ME405 board, the process of porting the libraries to function on the vehicle's board involves simple modification of parameters such as corresponding registers because the process to achieve something on the BNO055 remains the same across platforms.

The process in which the BNO055 communicates with our board is called I2C or Two Wire Interface which is supported by a wide range of systems. This process uses two lines or 'wires' to transmit data. The SCL wire is the clock which synchronizes the process and assures the correct speed of transmission. The SDA line is the data line which toggles from high to low to indicates numerous things from a start transmission command to the actual data being sent from Master to Slave or vice versa.

The custom driver created was named bno055\_driver in order to stay true to convention. Because this driver was created before the knowledge that a class i2c\_master driver was going to be provided, our group created our own custom i2c\_driver which is specialized for the BNO055 and based on the protocol described in the datasheet of the BNO055 IMU. i2c\_driver was written independently at first, but then adapted to include RTOS structure checks such as the handing off of the mutex to ensure that the i2c lines are not subjected to collision requests.

The bno055\_driver was based heavily on Programmer's Ktown's driver for the Arduino family of micro controllers and written for Adafruit Industries. Since the code was written in c++, compatibility was easy to achieve and much of his algorithms remain with slight alterations depending on the way we intended to use the information needed from the BNO055. Since we're only using EULER vectors, we saved space by only including a way to extract this data instead of making a driver which would cover a large range of uses.

Once testing was complete for bno055\_driver, we created three shares variables to hold our heading, pitch, and roll information. These variables would be continuously updated by a new task named task\_imu. We originally had this task running every 100ms with a task\_priority() of 2, however the i2c process was slowing the execution of this task. To fix it, we changed the way we returned information from the bno055\_driver's getVector() function and wrote specialized getHeading(), getPitch(), and getRoll() functions to only read two registers at a time instead of returning an extracted vector which involved reading all six at the same time.