A picture containing graphical user interface

Description automatically generated

ECE 595R

Robotics and Embedded Systems

Final Project

Bluetooth-Controlled Rover

Sensor Fusion and Kalman Filtering for Precision Motion

Matthew Thume

Instructor: Dr. Mirzaei

May 14th

Spring 2024

***Introduction***

This project integrates the MPU-6050 gyroscope and accelerometer (I2C) with the Adafruit Bluefruit LE (UART) to interface with the TI MSP432 Launchpad microcontroller. The goal was to enable Bluetooth-based control of the robot from a smartphone, with a focus on utilizing Kalman Filtering to process the MPU-6050’s axis data for precise rotational control. This allowed the robot to rotate by a user-specified number of degrees, achieving accurate motion control through the combined use of the sensor fusion.

***Components Used***

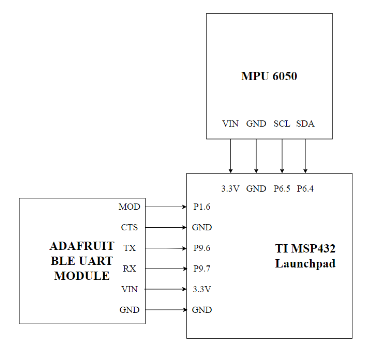
- TI MSP432 Launchpad

- TI-RSLK MAX Chassis

-Adafruit MPU-6050 3-Axis Accelerometer and Gyro Sensor

-Adafruit Bluefruit LE UART Friend

***Theory***The project integrates wireless communication and sensor technology into robotics by implementing a wireless communication module, along with a gyroscope and accelerometer for precise orientation and motion sensing. Figure 1 below presents a block diagram of the project’s connections.

  
Figure 1 - Project Block Diagram

The project utilizes the Adafruit Bluefruit LE UART Friend as a UART-to-Bluetooth Low Energy (BLE) bridge, facilitating data transmission between the robot's microcontroller and external devices. Operating via BLE, the Adafruit Bluefruit LE UART Friend ensures efficient data exchange with minimal power consumption with a maximum current draw of only 20mA. This module also enables simultaneous bidirectional within the project as well via the RX and TX lines. This is the major benefit of implementing a UART module as it allows for an on-demand command to be sent to the robot and still be able to receive our orientation data which is being continuously polled.

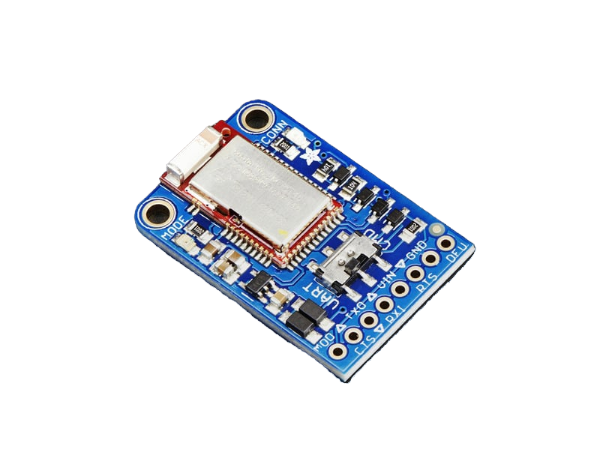


Figure 2 - Adafruit Bluefruit LE UART Friend Module

The MPU-6050 3-Axis Accelerometer and Gyro Sensor module provides the essential data for orientation and motion sensing. This module integrates both accelerometer and gyroscope functionalities, allowing us to capture precise movement and orientation data of the robot.

Gyroscopes measure angular velocity, providing information about the rate of rotation around each axis. Accelerometers, on the other hand, measure linear acceleration, indicating changes in velocity along each axis. By combining data from both sensors of the MPU-6050, we can accurately determine the robot's orientation in three-dimensional space and monitor its motion.

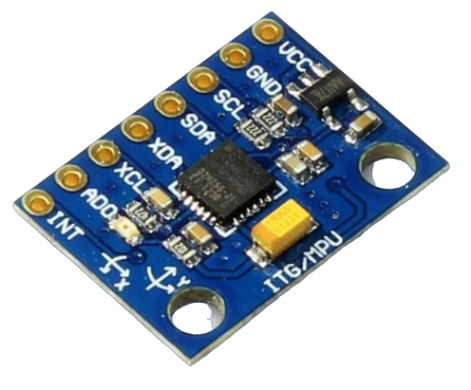


Figure 3 - MPU-6050 3-Axis Accelerometer and Gyro Sensor module

A visual representation of the orientation of the x, y, and z axes with respect to the module can be seen below in Figure 4. The visual representation helps to understand how the rotation of the module about the axes will be used to determine the calculation of the robot when mounted on board.

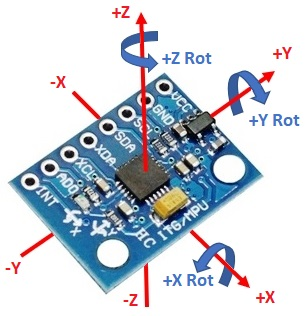


Figure 4 - MPU-6050 X-Y-Z-Axis Orientation

In order to process the sensor data to estimate the true orientation state of the system, a Kalman Filter was implemented to handle the Digital Motion Processing (DMP) through sensor fusion. Recall that the gyroscope data returns angular velocity while the accelerometer measures linear acceleration. Therefore with this data basically all we can determine is how fast we are rotating and in which direction. This raw gyroscope data alone does not provide an accurate orientation reading of our bot due to noise and bias. Therefore it was necessary to implement the Kalman Filter to address this issue.

The Kalman Filter algorithm combines information from multiple sensors to improve the accuracy of the estimated state. In this case, the Kalman Filter processes data from both the gyroscope and accelerometer of the MPU-6050 module. Through sensor fusion of the data received from both sensors, the Kalman Filter is able to calculate the orientation of the robot in three-dimensional space.

The Kalman Filter code implemented involves a one-dimensional Kalman Filter algorithm, which updates the state and uncertainty estimates based on the input and measurement data. This algorithm continuously polled to update the orientation estimates of the robot, providing smoother and more accurate orientation data for control and monitoring purposes.

Below is the code snippet for the Kalman Filter implementation:

| void kalman\_1d(float KalmanState,float KalmanUncertainty,float KalmanInput,... ...floatKalmanMeasurement)  {  KalmanState = KalmanState + 0.004 \* KalmanInput;  KalmanUncertainty = KalmanUncertainty + 0.004 \* 0.004 \* 4 \* 4;  float KalmanGain = KalmanUncertainty \* 1 / (1 \* KalmanUncertainty + 3 \* 3);  KalmanState = KalmanState + KalmanGain \* (KalmanMeasurement - KalmanState);  KalmanUncertainty = (1 - KalmanGain) \* KalmanUncertainty;  Kalman1DOutput[0] = KalmanState;  Kalman1DOutput[1] = KalmanUncertainty;  }  void gyro\_signals(void) {  float\* gyro\_data\_xyz = MPU\_6050\_Get\_Adjusted\_XYZ\_Gyroscope();  // Process gyro data  }  void Bot\_Orientation() {  gyro\_signals();  RateRoll -= RateCalibrationRoll;  RatePitch -= RateCalibrationPitch;  RateYaw -= RateCalibrationYaw;  kalman\_1d(KalmanAngleRoll, KalmanUncertaintyAngleRoll, RateRoll, AngleRoll);  KalmanAngleRoll = Kalman1DOutput[0];  KalmanUncertaintyAngleRoll = Kalman1DOutput[1];  kalman\_1d(KalmanAnglePitch, KalmanUncertaintyAnglePitch, RatePitch, AnglePitch);  KalmanAnglePitch = Kalman1DOutput[0];  KalmanUncertaintyAnglePitch = Kalman1DOutput[1];  kalman\_1d(KalmanAngleYaw, KalmanUncertaintyAngleYaw, RateYaw, AngleYaw);  KalmanAngleYaw = Kalman1DOutput[0];  KalmanUncertaintyAngleYaw = Kalman1DOutput[2];  //iPhone RX  char ble\_data[50];  sprintf(ble\_data, "R: %.2f, P: %.2f, YAW: %.2f", KalmanAngleRoll, KalmanAnglePitch,KalmanAngleYaw);  BLE\_UART\_OutString(ble\_data);    //serial monitor  printf("Roll Angle [] %.2f Pitch Angle [] %.2f\n YAW Angle [] %.2f\n", KalmanAngleRoll, KalmanAnglePitch,KalmanAngleYaw);  } |
| --- |

The Kalman Filter code processes data from the gyroscope and accelerometer sensors to accurately estimate the pitch, roll, and yaw angles of the robot, enabling precise determination of its orientation. With this implementation, we've introduced the capability to wirelessly control the robot's orientation, allowing users to send turn commands to achieve specific degrees of rotation.

As the Kalman Filter continuously works to create an accurate calculation of the bot's rotation, a function RotateBot() was created to handle the user specified turn angle. Within RotateBot() an iterative process of monitoring and adjusting orientation until the desired orientation is achieved. .

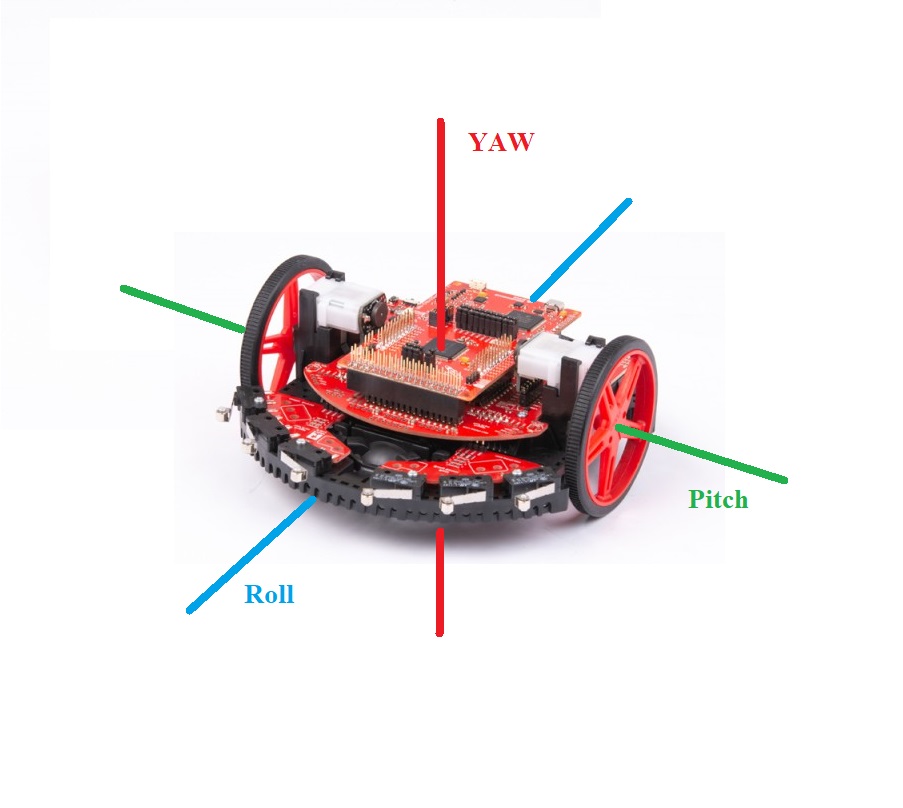


Figure 5 - Visual Axis Orientation Representation

***Analysis and Results***

1. **Command Implementation**

Implementation of all the commands was simple, read from the UART buffer and use a series of *if else* statements to compare the command sent to the ones we have predefined. Once the command sent is verified as being valid it will perform the operation of the given command. In Figure 6 is a list of all of our predefined commands which was sent to the robot using UART protocol. There is room for improvement with how the robot reacts to each command as well as adding more commands in the future. For example, the “F” command can be modified so that the user can determine the distance that the robot will travel. This can be easily implemented since the MPU 6050 has an accelerometer.

| Command | Operation |
| --- | --- |
| F | Move Forward |
| S | Stop |
| B | Move Backwards |
| L (x) | Rotate Left x Degrees |
| R (x) | Rotate Right x Degrees |

Figure 6 - UART Command List

1. **Rotation**

The idea for implementing the degree of rotation was originally to constantly poll the orientation of the robot. A periodic interrupt was originally used to call the Bot\_Orientation function, which would give us a real time orientation of the robot, but this caused two issues. First, due to the intense amount of calculations that is done every time a call is made to Bot\_Orientation, whenever a command was sent the robot would not respond. The team found that the data in the UART buffer was not able to be processed before the next interrupt occurred. To try and resolve this issue the time in between each interrupt was lengthened to try and account for the processing time but the issue remained. The second issue that the team found was that due to the natural drift of the gyroscope if we constantly polled the orientation we would not be able to get an accurate reading of its angle relative to its starting position since it would register a change in angle even while standing still.

At this point, it was decided to move in a different direction with the code. A RotateBot function was added, which took two parameters: direction and angle. The function determines the direction of rotation and starts the motors. The orientation is then polled until the desired angle is reached, at which point the motors stop. This solved the interrupt issue since the interrupt had been completely removed, and the orientation is only being polled while the robot is rotating. The gyroscope drift issue still remained though. It was then noticed that the gyroscope would only drift while sitting still, if the robot was rotating at a constant rate, the angular readings were accurate. Taking this into account, the orientation angle of the robot would be reset back to zero at the beginning of the function call to RotateBot. At this point all of the issues were resolved, and the bot was able to perform the first accurate rotation.

***Conclusion***

The project effectively integrates wireless communication and sensor fusion to achieve precise orientation and motion sensing through Digital Motion Processing. Through the implementation of the Adafruit Bluefruit LE UART Friend and the MPU-6050 3-Axis Accelerometer and Gyro Sensor module, we enable efficient data exchange and accurate estimation of the robot's orientation in three-dimensional space. The incorporation of the Kalman Filter algorithm further enhances orientation accuracy, opening the door for user-specified turn commands via the wireless control functionality of the robot. Our system ensures iterative monitoring and adjustment of orientation until the desired angle is achieved. A real-time 3D visualization of the bots orientation or more data acquisition sensors can be implemented in the future furthering the capabilities of the bot’s functionality.