THRUST VECTOR CONTROL

Krutik Shah

Embedded Software Design, Final Project

AGENDA

System Introduction

Context Diagram

Sensors & Components

Controller Interfaces

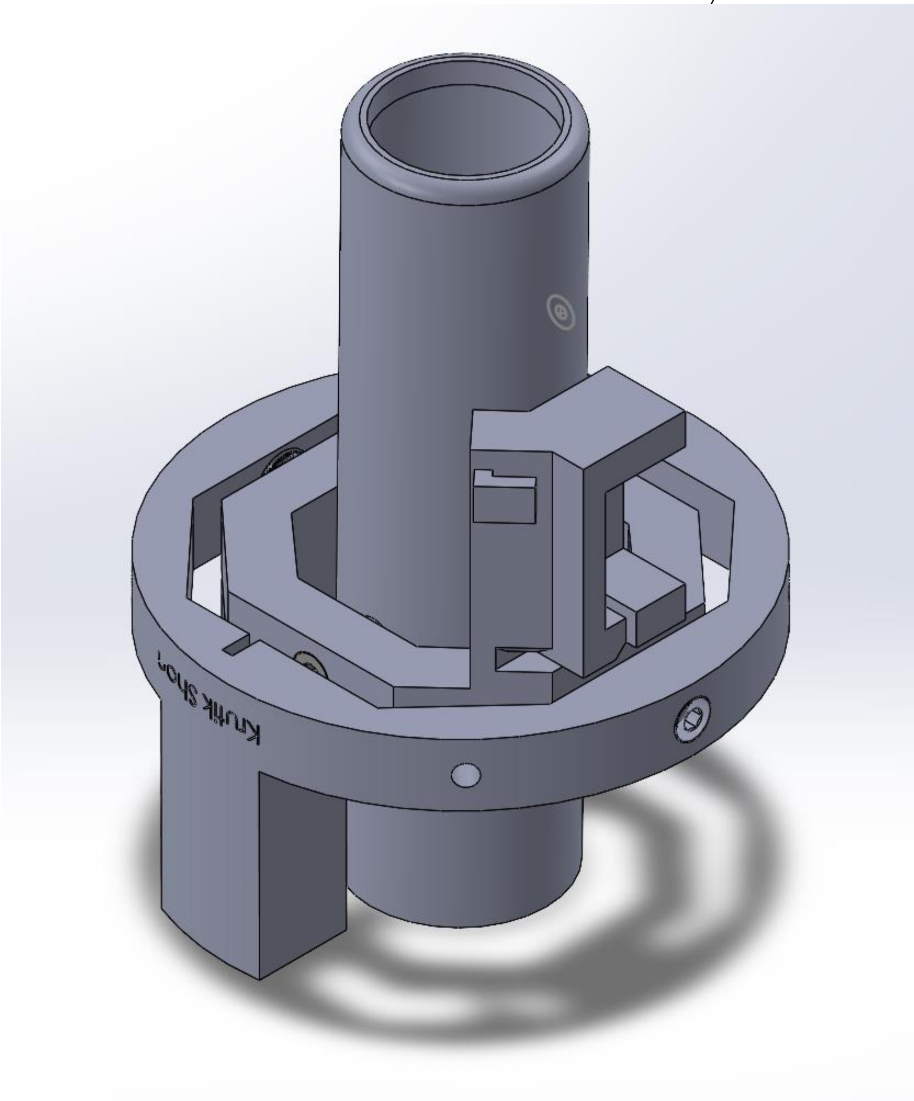
MBED Drivers/Constructs

Code

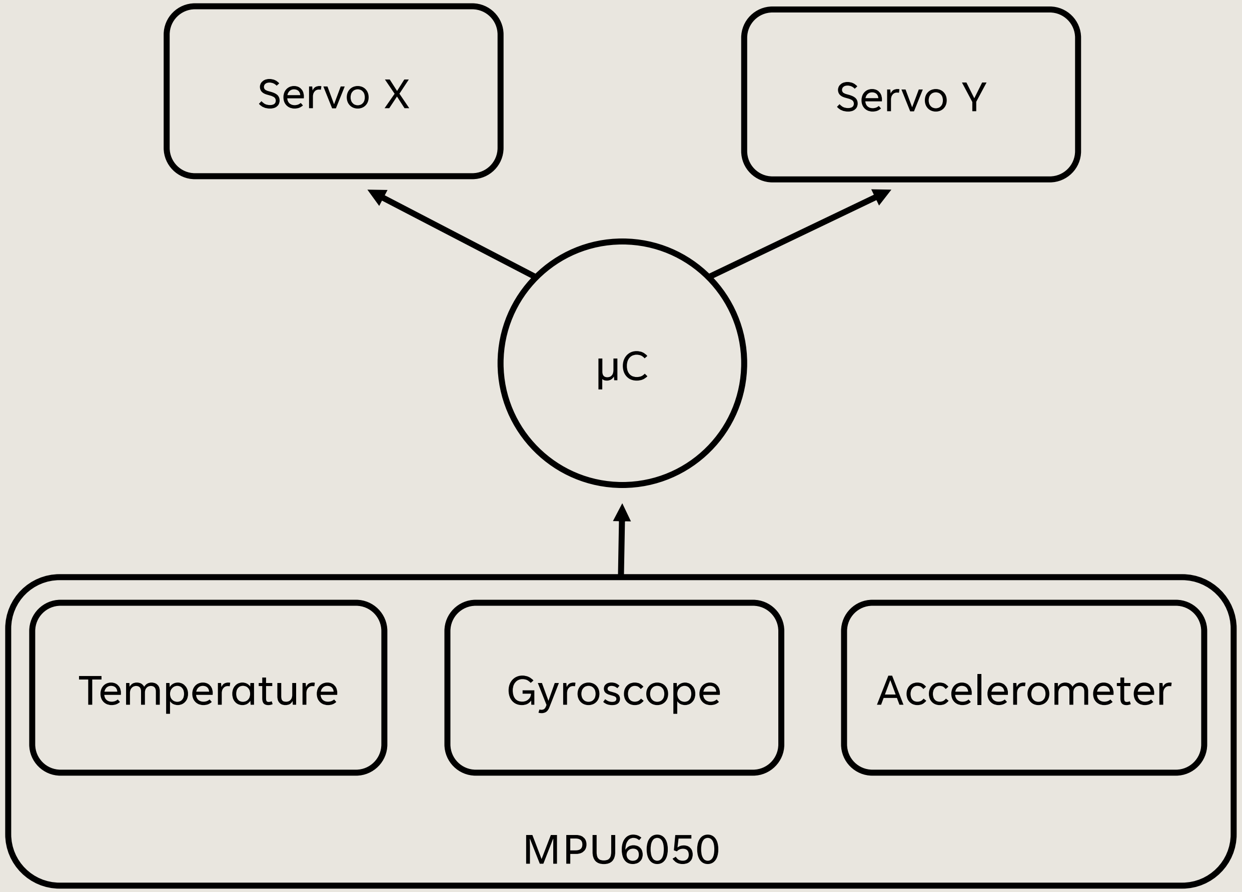
Problems & Improvements

SYSTEM INTRODUCTION

- Thrust Vector Control is a system used in rockets
- Stabilizes trajectory
- Most commonly a closed-loop control system utilizing PID control
- Collects angle of rocket motor, move servos based on that
- Additionally collects temperature readings and accelerometer readings

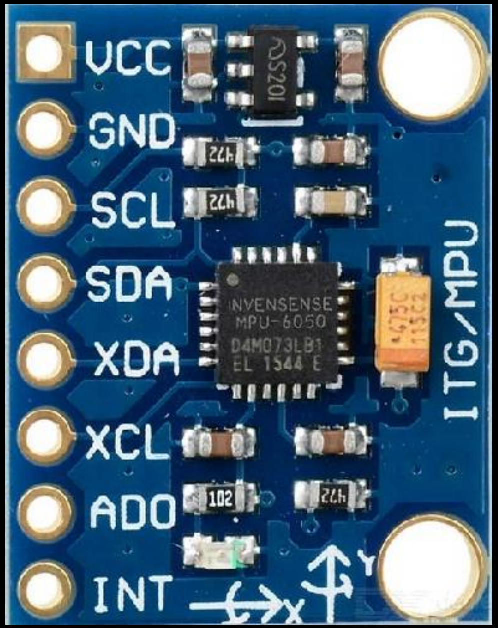


CONTEXT DIAGRAM



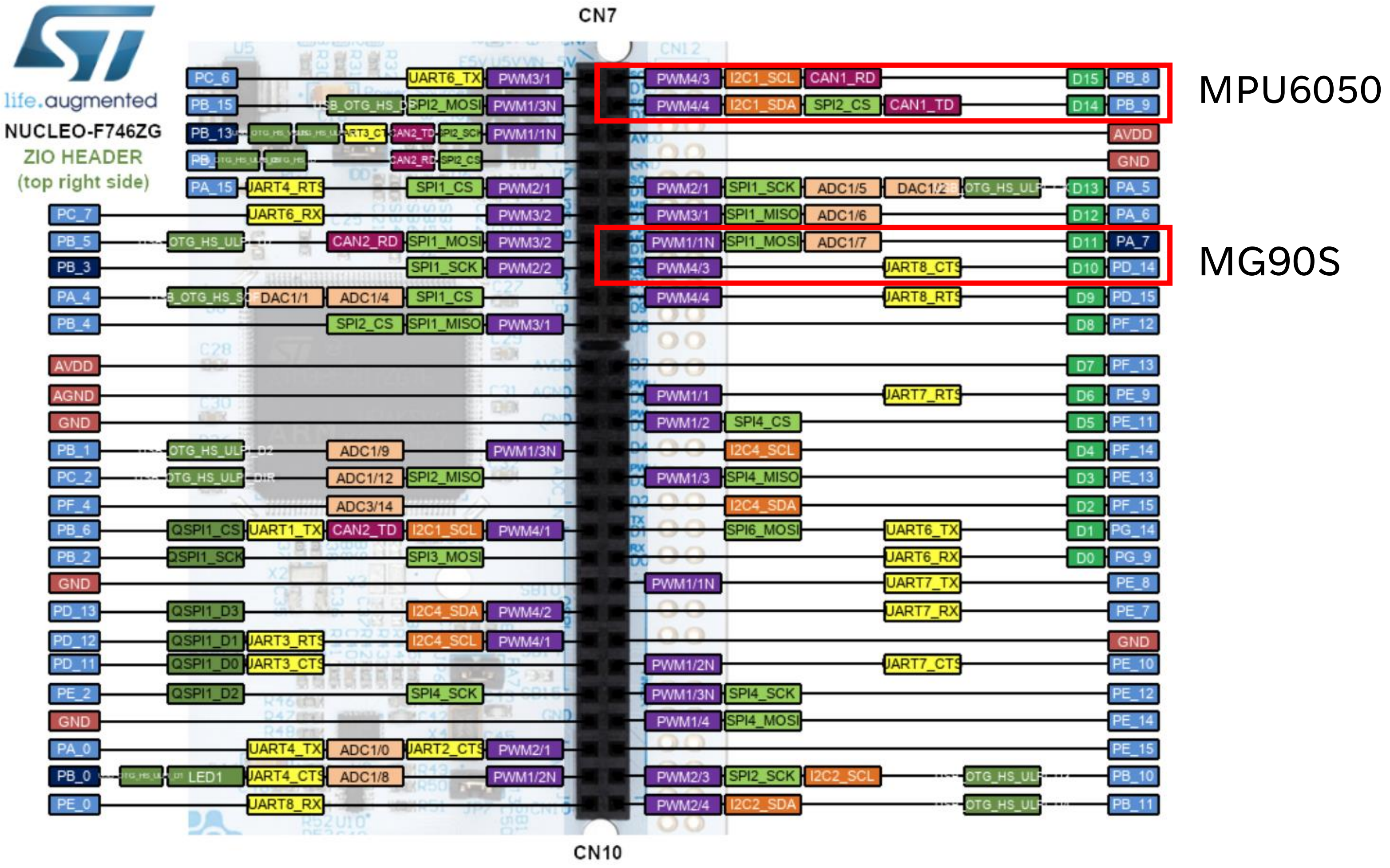
SENSORS & COMPONENTS

- MPU6050
  - 3-axis gyroscope, 3-axis accelerometer, temperature sensor
- 2x MG90S servos
  - For X & Y axes of movement



CONTROLLER INTERFACES

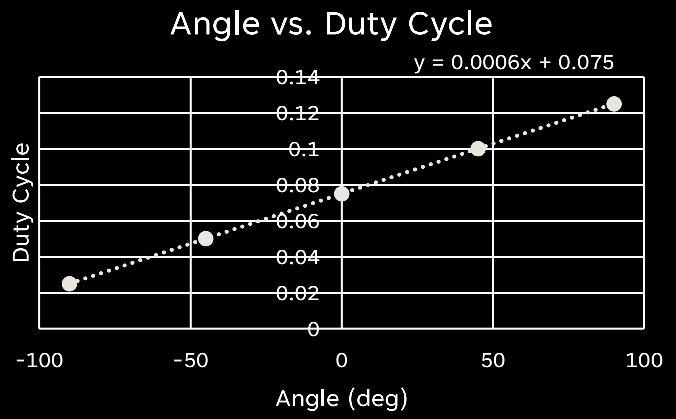
- MPU6050 – I^{2}C
  - pins D14 (SDA) and D15 (SCL)
- MG90S – PWM
  - Servo X – pin D10
  - Servo Y – pin D11



MBED CONSTRUCTS, LIBRARIES

- Timer
    - Used to calculate angle
    - Gyroscope outputs rad/s, so timer gets elapsed time
- PWM for servos
- MPU6050 Library

CODE




```
void runServo(float angleXAvg, float angleYAvg)
{
    //TVC Startup//
    dutyCycleA = (0.0006 * angleYAvg) + 0.075; //using equation from Angle vs. Duty Cycle graph
    dutyCycleB = (0.0006 * angleYAvg) + 0.075;
    servoB.write(dutyCycleB);
    servoA.write(dutyCycleA);
    printf("duty cycle A: %f    | duty cycle B: %f\n", dutyCycleA, dutyCycleB);
    wait_us(WAIT);
}
```

```
float gyro[3];
//Collecting gyro angles three times and averaging them for more accurate results
for (int i = 0; i < 2; i++)
{
    mpu.getGyro(gyro);
    angleX = rad_to_deg(gyro[0]) * elapsedTime;
    angleY = rad_to_deg(gyro[1]) * elapsedTime;
    angleZ = rad_to_deg(gyro[2]) * elapsedTime;

    if (i == 0)
    {
        angleXSample1 = angleX;
        angleYSample1 = angleY;
        angleZSample1 = angleZ;
    }
    else if (i == 1)
    {
        angleXSample2 = angleX;
        angleYSample2 = angleY;
        angleZSample2 = angleZ;
    }
    else if (i == 2)
    {
        angleXSample3 = angleX;
        angleYSample3 = angleY;
        angleZSample3 = angleZ;
    }
}

//averaging sample gyro data
angleXAvg = (angleXSample1 + angleXSample2 + angleXSample3) / 3;
angleYAvg = (angleYSample1 + angleYSample2 + angleYSample3) / 3;
angleZAvg = (angleZSample1 + angleZSample2 + angleZSample3) / 3;
```

```
//setup, test connection with gyro and stop program if bad connection
mpu.setSleepMode(false);
bool test = mpu.testConnection();
if (test == true)
{
    printf("Connection: Good\n");
}
else if (test == false) {
    printf("Connection: Bad\n");
    return(0);
}
```

```
//temperature readings
float temp = mpu.getTemp();
printf("temprature = %0.2f ^C\r\n",temp);
```

```
//accelerometer data
float acce[3];
mpu.getAccelero(acce);
accelX = rad_to_deg(acce[0]);
accelY = rad_to_deg(acce[1]);
accelZ = rad_to_deg(acce[2]);
printf("AccelX=%f,  AccelY=%f,  AccelZ=%f  (m/s^2)\r\n",acce[0],acce[1],acce[2]);
```

RESULTS

```
Connection: Good
temprature = 24.11 ^C
Angle X: 9.985065     | Angle Y: -3.360016   | Angle Z: -0.474914
duty cycle A: 0.072984    | duty cycle B: 0.072984
AccelX=-9.536968,  AccelY=-0.905317,  AccelZ=-0.407153  (m/s^2)
Connection: Good
temprature = 23.92 ^C
Angle X: 9.725629     | Angle Y: -3.222097   | Angle Z: -0.907884
duty cycle A: 0.073067    | duty cycle B: 0.073067
AccelX=-9.539363,  AccelY=-0.862207,  AccelZ=-0.502954  (m/s^2)
Connection: Good
temprature = 23.97 ^C
Angle X: 9.753545     | Angle Y: -3.241287   | Angle Z: -0.605515
duty cycle A: 0.073055    | duty cycle B: 0.073055
AccelX=-9.572893,  AccelY=-0.907712,  AccelZ=-0.426313  (m/s^2)
Connection: Good
temprature = 24.06 ^C
Angle X: 9.695959     | Angle Y: -3.489121   | Angle Z: -1.133371
duty cycle A: 0.072907    | duty cycle B: 0.072907
AccelX=-9.730965,  AccelY=-0.874182,  AccelZ=-0.428708  (m/s^2)
Connection: Good
temprature = 23.97 ^C
Angle X: 9.967256     | Angle Y: -3.360016   | Angle Z: -1.015128
duty cycle A: 0.072984    | duty cycle B: 0.072984
AccelX=-9.462722,  AccelY=-0.821492,  AccelZ=-0.428708  (m/s^2)
```

CHALLENGES

- Libraries available for the MPU6050 sensor
- Displaying gyroscope data
- More fine error-correcting movement

FUTURE IMPROVEMENTS

- Utilize PID control for better error-correcting movement
- Use a wireless interface to receive sensor data remotely
- Custom PCB, can output to SD card a data log of sensor data throughout full flight