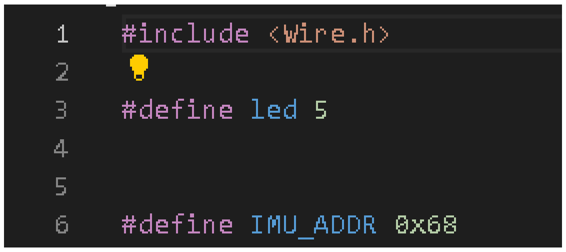
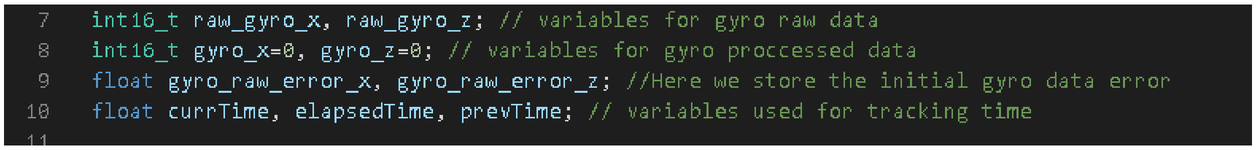
Rons broomstick



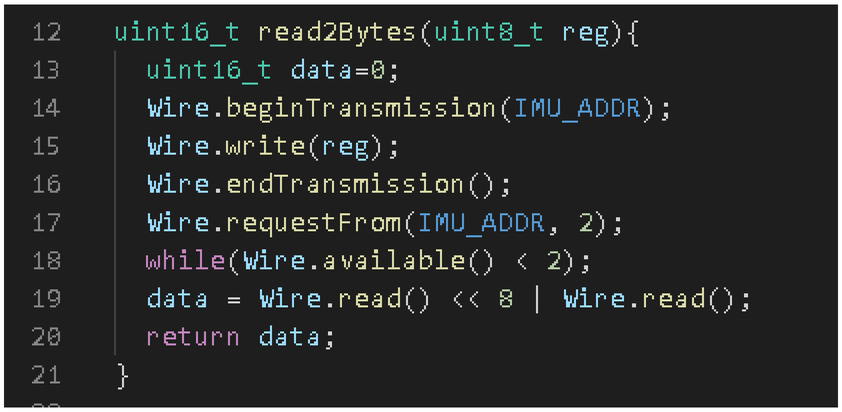
```
#include <Wire.h>

#define led 5


#define IMU_ADDR 0x68
```

We included wire.h because it’s a very important library for the communication with the IMU. The IMU_ADDR is the address of the IMU to be able to communicate.



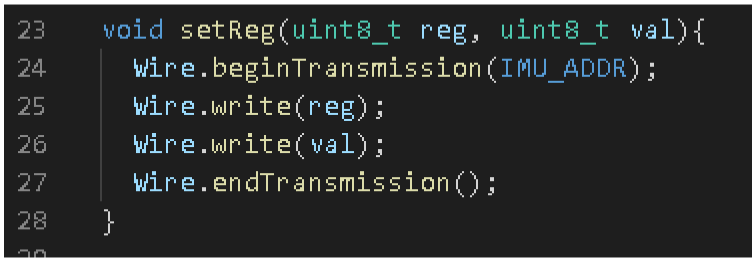
```
int16_t raw_gyro_x, raw_gyro_z; // variables for gyro raw data
int16_t gyro_x=0, gyro_z=0; // variables for gyro proccessed data
float gyro_raw_error_x, gyro_raw_error_z; //Here we store the initial gyro data error
float currTime, elapsedTime, prevTime; // variables used for tracking time
```

The important variables that will be used throughout the program t oread raw data and to calculate the angles. We only needed x and z for this project as we’re looking for the pitch and roll angles.



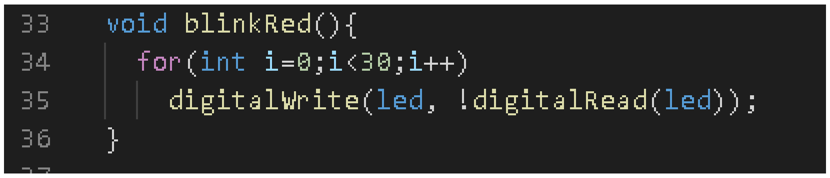
```
uint16_t read2Bytes(uint8_t reg){
  uint16_t data=0;
  Wire.beginTransmission(IMU_ADDR);
  Wire.write(reg);
  Wire.endTransmission();
  Wire.requestFrom(IMU_ADDR, 2);
  while(Wire.available() < 2);
  data = Wire.read() << 8 | Wire.read();
  return data;
}
```

The read2Bytes function reads two bytes from the IMU given a registry by reading 1 byte then shifting it 1 byte to the left then reading another byte.



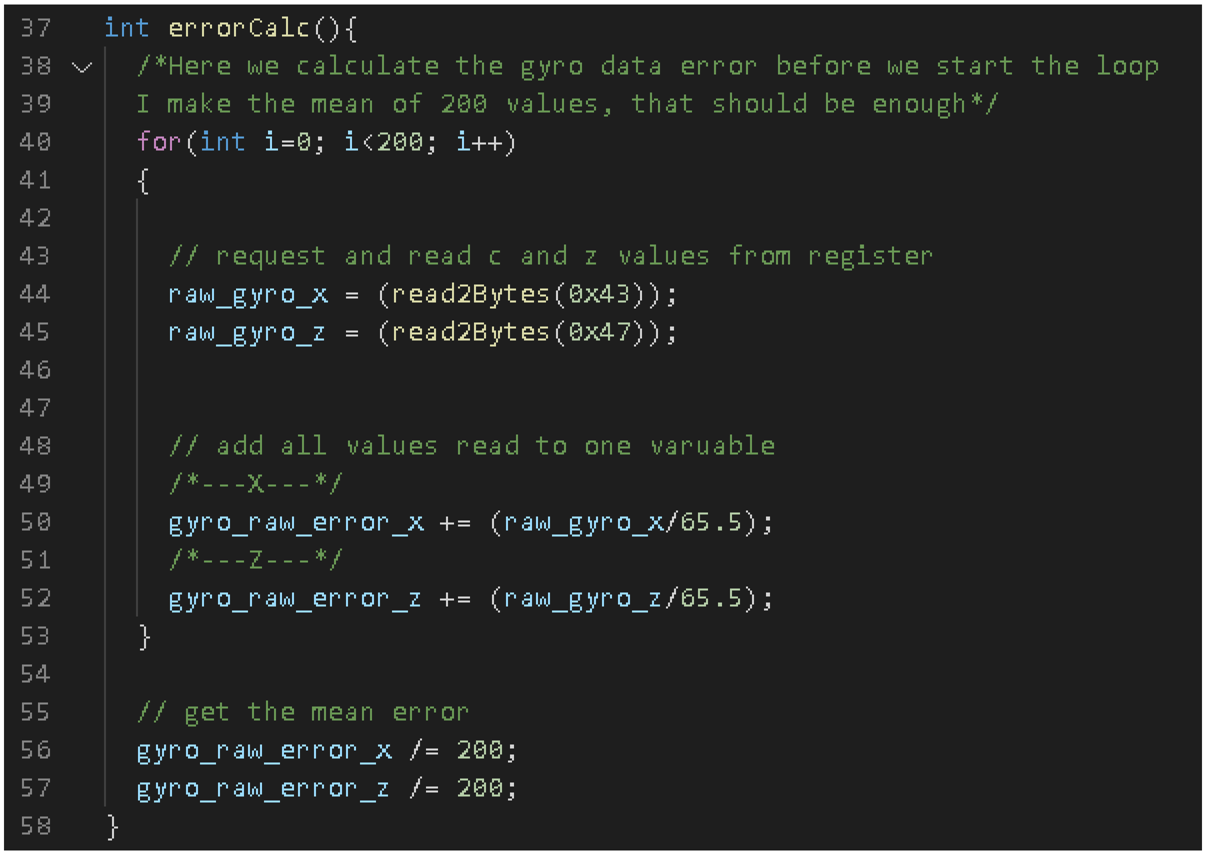
```
void setReg(uint8_t reg, uint8_t val){
  Wire.beginTransmission(IMU_ADDR);
  Wire.write(reg);
  Wire.write(val);
  Wire.endTransmission();
}
```

This function is used for setting registry configs. It was only used once for setting the gyro config.



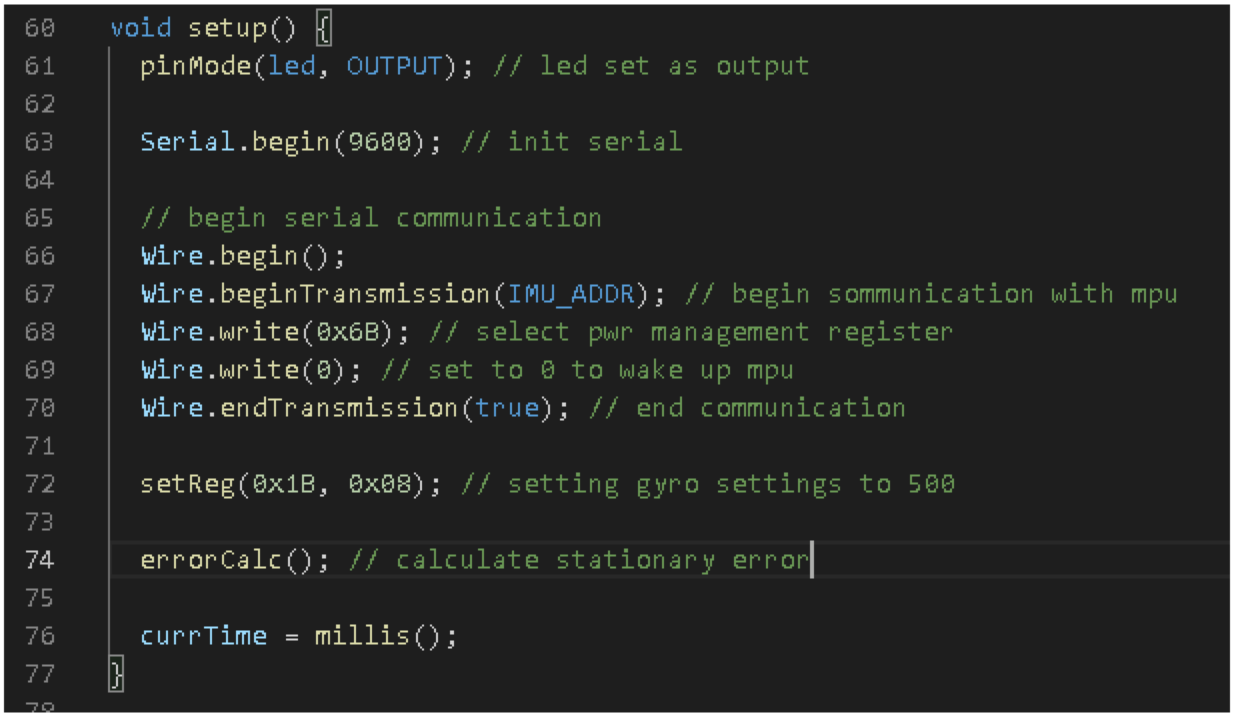
```
void blinkRed(){
  for(int i=0;i<30;i++)
    digitalWrite(led, !digitalRead(led));
}
```

Simple blink method that changes the state of the led.



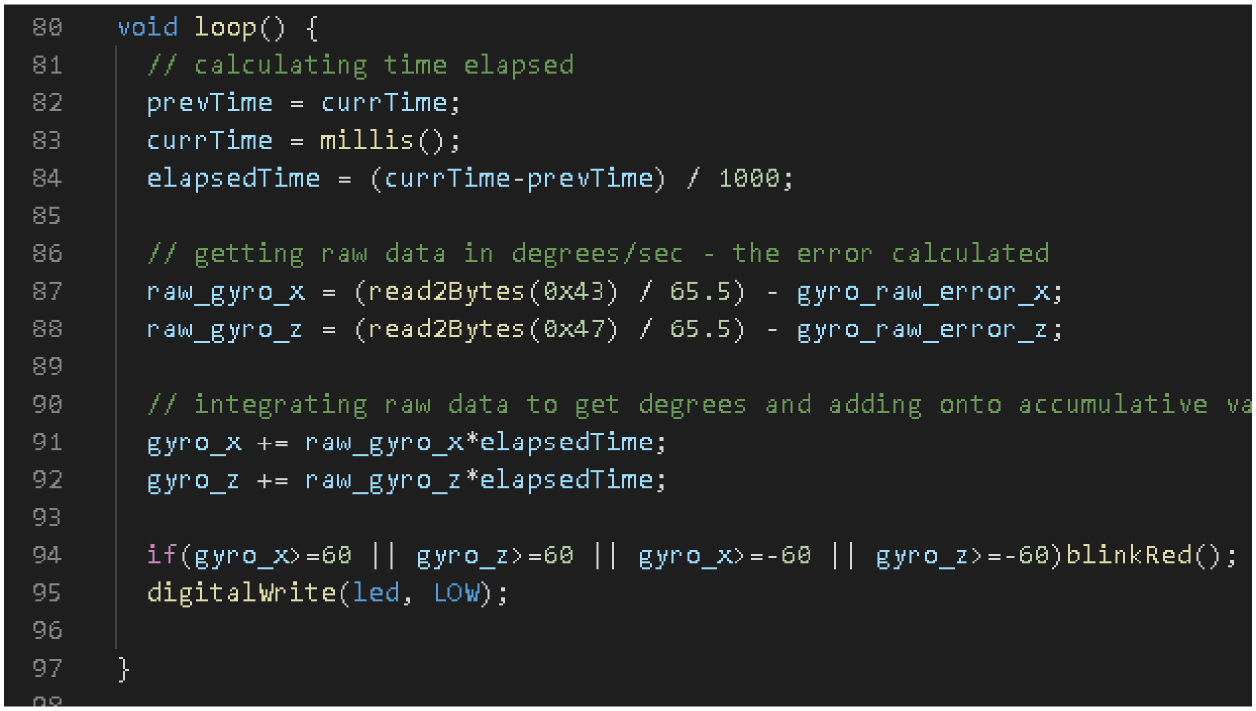
```
int errorCalc(){
  /*Here we calculate the gyro data error before we start the loop
  I make the mean of 200 values, that should be enough*/
  for(int i=0; i<200; i++)
  {

    // request and read c and z values from register
    raw_gyro_x = (read2Bytes(0x43));
    raw_gyro_z = (read2Bytes(0x47));


    // add all values read to one varuable
    /*---X---*/
    gyro_raw_error_x += (raw_gyro_x/65.5);
    /*---Z---*/
    gyro_raw_error_z += (raw_gyro_z/65.5);
  }

  // get the mean error
  gyro_raw_error_x /= 200;
  gyro_raw_error_z /= 200;
}
```

Error calculation function that should only be used when the sensor is stationary. It reads the stationary values 200 times then gets the mean of that raw data and that would be the error value that should be removed from our values later on.



```
void setup() {
  pinMode(led, OUTPUT); // led set as output

  Serial.begin(9600); // init serial

  // begin serial communication
  Wire.begin();
  Wire.beginTransmission(IMU_ADDR); // begin sommunication with mpu
  Wire.write(0x6B); // select pwr management register
  Wire.write(0); // set to 0 to wake up mpu
  Wire.endTransmission(true); // end communication

  setReg(0x1B, 0x08); // setting gyro settings to 500

  errorCalc(); // calculate stationary error

  currTime = millis();
}
```

In the setup we set the pin mode for the led and begin transmission with the IMU to wake it up from its default sleep state. The we set the gyro config to 500 dps. Then we calculate stationary error and finally get the current time.



```
void loop() {
  // calculating time elapsed
  prevTime = currTime;
  currTime = millis();
  elapsedTime = (currTime-prevTime) / 1000;

  // getting raw data in degrees/sec - the error calculated
  raw_gyro_x = (read2Bytes(0x43) / 65.5) - gyro_raw_error_x;
  raw_gyro_z = (read2Bytes(0x47) / 65.5) - gyro_raw_error_z;

  // integrating raw data to get degrees and adding onto accumulative va
  gyro_x += raw_gyro_x*elapsedTime;
  gyro_z += raw_gyro_z*elapsedTime;

  if(gyro_x>=60 || gyro_z>=60 || gyro_x>=-60 || gyro_z>=-60)blinkRed();
  digitalWrite(led, LOW);

}
```

In the loop, we first calculate elapsed time since last calculation. Then we get raw data and convert it to degrees per second and subtract the previously calculated error. Then, we convert it to degrees by integrating it or multiplying it by the elapsed time. Finally, we check if either angle has accumulated to more that 60 in which case, we rapidly brink red led and then start loop again.