////”FINAL CODE”/////

#include <Wire.h>

#include <SPI.h>

#include <SparkFunLSM9DS1.h>

#include "SensorFusion.h"

#include <Adafruit\_GFX.h>

#include <Adafruit\_ILI9341.h>

#define TFT\_DC 9

#define TFT\_RST 8 // You can also connect this to the Arduino reset in which case, set this #define pin to -1!

#define TFT\_CS 10

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC, TFT\_RST);

#define GRAVITY 9.81

#define PRINT\_INTERVAL 12

unsigned long lastPrint = 0; // Keep track of print time

LSM9DS1 imu;

SF filter;

float Axyz[3], Mxyz[3], Gxyz[3];

float pitch, roll, yaw;

float deltat;

void twoDots(float angle1, float angle2, int hue, int sat){

int center = tft.width()/4;

int radius = tft.width()/8;

int move = map(angle1, -90,90, radius, tft.height()-radius);

tft.fillCircle(center, move, radius, 0x07E0);

tft.fillCircle(tft.width()-center, tft.height()-move, radius, 0x07E0);

}

void setup() {

//LEVEL BAR

tft.begin();

tft.setRotation(1);

tft.fillScreen(0x0000);

tft.println("Level Bar Project");

tft.drawFastHLine(0, tft.height()/2, tft.width(), 0x8410);

tft.drawFastVLine(tft.width()/2, 0, tft.height(), 0x8410);

# //LEVEL BAR

pinMode(LED\_BUILTIN, OUTPUT);

Serial.begin(115200);

while (!Serial){}

Wire1.begin();

delay(100);

if (imu.begin(0x6B, 0x1E, Wire1)==false) {

while (1) {

digitalWrite(LED\_BUILTIN, HIGH);

delay(10);

digitalWrite(LED\_BUILTIN, LOW);

delay(20);

}

}

for (int i = 0; i < 5; i++) {

digitalWrite(LED\_BUILTIN, HIGH);

delay(100);

digitalWrite(LED\_BUILTIN, LOW);

delay(50);

}

}

void loop() {

if (imu.accelAvailable() && imu.magAvailable() && imu.gyroAvailable()) {

imu.readAccel();

imu.readMag();

imu.readGyro();

get\_IMU(Axyz, Mxyz, Gxyz);

Gxyz[0]= imu.calcGyro(imu.gx) \* DEG\_TO\_RAD;

Gxyz[1]= imu.calcGyro(imu.gy) \* DEG\_TO\_RAD;

Gxyz[2]= imu.calcGyro(imu.gz) \* DEG\_TO\_RAD;

Axyz[0] = imu.calcAccel(Axyz[0]) \* GRAVITY;

Axyz[1] = imu.calcAccel(Axyz[1]) \* GRAVITY;

Axyz[2] = imu.calcAccel(Axyz[2]) \* GRAVITY;

Mxyz[0] = imu.calcMag(Mxyz[0]);

Mxyz[1] = imu.calcMag(Mxyz[1]);

Mxyz[2] = imu.calcMag(Mxyz[2]);

deltat = filter.deltatUpdate();

filter.MadgwickUpdate( -Gxyz[0], Gxyz[1], Gxyz[2], // Flip Gyro Handedness

-Axyz[0], Axyz[1], Axyz[2], // Flip Accel Handedness

Mxyz[0], Mxyz[1], Mxyz[2], deltat);

/\*filter.MahonyUpdate( -Gxyz[0], Gxyz[1], Gxyz[2], // Flip Gyro Handedness

-Axyz[0], Axyz[1], Axyz[2], // Flip Accel Handedness

Mxyz[0], Mxyz[1], Mxyz[2], deltat);\*/

if (millis() - lastPrint >= PRINT\_INTERVAL) {

lastPrint = millis();

Serial.print("Orientation: ");

Serial.print((int)filter.getYaw());

Serial.print(", ");

Serial.print((int)filter.getPitch());

Serial.print(", ");

Serial.println((int)filter.getRoll());

twoDots((int)filter.getYaw(), 120, 0, 100); //too slow, displays both dots

//==================================================

// Debugging code

//==================================================

/\*

Serial.print("A: ");

Serial.print(-Axyz[0]);

Serial.print(", ");

Serial.print(Axyz[1]);

Serial.print(", ");

Serial.println(Axyz[2]);

Serial.print("G: ");

Serial.print(-Gxyz[0]);

Serial.print(", ");

Serial.print(Gxyz[1]);

Serial.print(", ");

Serial.println(Gxyz[2]);

Serial.print("M: ");

Serial.print(Mxyz[0]);

Serial.print(", ");

Serial.print(Mxyz[1]);

Serial.print(", ");

Serial.println(Mxyz[2]);

float \* q;

q = filter.getQuat();

Serial.print("Quaternion: ");

Serial.print(q[0], 4);

Serial.print(", ");

Serial.print(q[1], 4);

Serial.print(", ");

Serial.print(q[2], 4);

Serial.print(", ");

Serial.println(q[3], 4);

\*/

}

}

}

//==================================================

// Sensor calibration

//==================================================

// Gyroscope scale 6194.6 to normalize

float G\_B[3] {220.78, -262.23, 254.91};

float G\_Ainv[3][3] {{ 0.88241, -0.01470, -0.00232},

{ -0.01470, 1.03994, 0.00201},

{ -0.00232, 0.00201, 1.08761}};

// Accelerometer scale 16477.7 to normalize

float A\_B[3] {-247.51, -81.74, 316.41};

float A\_Ainv[3][3] {{ 1.00076, 0.00025, -0.00001},

{ 0.00025, 1.00251, -0.00667},

{ -0.00001, -0.00667, 1.00254}};

// Magnetometer scale 3936.7 to normalize

float M\_B[3] {21.74, -129.71, -3045.94};

float M\_Ainv[3][3] {{ 1.48159, 0.05738, -0.00065},

{ 0.05738, 1.48909, -0.00508},

{ -0.00065, -0.00508, 1.53282}};

//==================================================

// Apply calibration bias offset and scale factors

//==================================================

void get\_IMU(float Axyz[3], float Mxyz[3], float Gxyz[3]) {

byte i;

float temp[3];

Axyz[0] = imu.ax;

Axyz[1] = imu.ay;

Axyz[2] = imu.az;

Mxyz[0] = imu.mx;

Mxyz[1] = imu.my;

Mxyz[2] = imu.mz;

Gxyz[0] = imu.gx;

Gxyz[1] = imu.gy;

Gxyz[2] = imu.gz;

// Gyroscope

for (i = 0; i < 3; i++) temp[i] = Gxyz[i] - G\_B[i];

Gxyz[0] = G\_Ainv[0][0] \* temp[0] + G\_Ainv[0][1] \* temp[1] + G\_Ainv[0][2] \* temp[2];

Gxyz[1] = G\_Ainv[1][0] \* temp[0] + G\_Ainv[1][1] \* temp[1] + G\_Ainv[1][2] \* temp[2];

Gxyz[2] = G\_Ainv[2][0] \* temp[0] + G\_Ainv[2][1] \* temp[1] + G\_Ainv[2][2] \* temp[2];

// Accelerometer

for (i = 0; i < 3; i++) temp[i] = Axyz[i] - A\_B[i];

Axyz[0] = A\_Ainv[0][0] \* temp[0] + A\_Ainv[0][1] \* temp[1] + A\_Ainv[0][2] \* temp[2];

Axyz[1] = A\_Ainv[1][0] \* temp[0] + A\_Ainv[1][1] \* temp[1] + A\_Ainv[1][2] \* temp[2];

Axyz[2] = A\_Ainv[2][0] \* temp[0] + A\_Ainv[2][1] \* temp[1] + A\_Ainv[2][2] \* temp[2];

// Magnetometer

for (i = 0; i < 3; i++) temp[i] = Mxyz[i] - M\_B[i];

Mxyz[0] = M\_Ainv[0][0] \* temp[0] + M\_Ainv[0][1] \* temp[1] + M\_Ainv[0][2] \* temp[2];

Mxyz[1] = M\_Ainv[1][0] \* temp[0] + M\_Ainv[1][1] \* temp[1] + M\_Ainv[1][2] \* temp[2];

Mxyz[2] = M\_Ainv[2][0] \* temp[0] + M\_Ainv[2][1] \* temp[1] + M\_Ainv[2][2] \* temp[2];

}

/////TEST BENCH/////

#include <Adafruit\_ILI9341.h>

//#include <ILI9341\_Fast.h>

#include <Adafruit\_GFX.h> // Include core graphics library

// Declare pins for the display:

#define TFT\_DC 9

#define TFT\_RST 8 // You can also connect this to the Arduino reset in which case, set this #define pin to -1!

#define TFT\_CS 10

// Create display:

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC, TFT\_RST);

static const unsigned char PROGMEM InvaderGFX []={

B00100000, B10000000,

B00010001, B00000000,

B00111111, B10000000,

B01101110, B11000000,

B11111111, B11100000,

B10111111, B10100000,

B10100000, B10100000,

B00011011, B00000000

};

int XPos = 0;

int YPos = 0;

void setup() {

tft.begin();

tft.setRotation(1);

tft.fillScreen(0x0000); //black screen

//tft.fillCircle(80, 120, 30, 0x07E0); //x,y,radius,color 320 x 240

//tft.fillCircle(240, 120, 30, 0x07E0); //x,y,radius,color 320 x 240

tft.println("Level Bar Project");

}

void loop() {

//tft.clearScreen();

tft.drawFastHLine(0, tft.height()/2, tft.width(), 0x8410); //figure out how to clear tft display

tft.drawFastVLine(tft.width()/2, 0, tft.height(), 0x8410);

//draw prev position cirle

//cover prev positin circle in black

//new circle in color

tft.drawCircle(XPos, 120, 30, 0x07E0); //x,y,radius,color 320 x 240

tft.drawCircle(XPos, 120, 30, 0x0000); //x,y,radius,color 320 x 240

tft.drawCircle(XPos+1, 120, 30, 0x07E0); //x,y,radius,color 320 x 240

XPos+=1;

if(XPos>320)

XPos=0;

}

void twoDots(float angle1, float angle2, int hue, int sat){

int center = tft.width()/4;

int radius = tft.width()/8;

int move = map(angle1, -90,90, radius, tft.height()-radius);

tft.fillCircle(center, move, radius, 0x07E0);

tft.fillCircle(tft.width()-center, tft.height()-move, radius, 0x07E0);

}

///Placeholder///

#include <Wire.h>

#include <SPI.h>

#include <SparkFunLSM9DS1.h>

#include "SensorFusion.h"

#include <Adafruit\_GFX.h>

#include <Adafruit\_ILI9341.h>

#define TFT\_DC 9

#define TFT\_RST 8 // You can also connect this to the Arduino reset in which case, set this #define pin to -1!

#define TFT\_CS 10

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC, TFT\_RST);

#define GRAVITY 9.81

#define PRINT\_INTERVAL 12

unsigned long lastPrint = 0; // Keep track of print time

LSM9DS1 imu;

SF filter;

float Axyz[3], Mxyz[3], Gxyz[3];

int pitch, roll, yaw;

float deltat;

float yeehaw=0;

void twoDots(float angle1, float angle2, int hue, int sat){

int center = tft.width()/4;

int radius = 15; //tft.width()/8

int move = map(angle1, -90,90, radius, tft.height()-radius);

//draw prev position cirle

//cover prev positin circle in black

//new circle in color

tft.drawCircle(center, move, radius, 0x07E0); //circle 1

tft.drawCircle(tft.width()-center, tft.height()-move, radius, 0x07E0); //circle 1, x axis, y axis, size in radius, color

tft.drawCircle(center, move, radius, 0x0000); //circle 2

tft.drawCircle(tft.width()-center, tft.height()-move, radius, 0x0000); //circle 2, x axis, y axis, size in radius, color

tft.drawFastHLine(0, tft.height()/2, tft.width(), 0x8410);

tft.drawFastVLine(tft.width()/2, 0, tft.height(), 0x8410);

}

void setup() {

//LEVEL BAR

tft.begin();

tft.setRotation(1);

tft.fillScreen(0x0000);

tft.println("Level Bar Project");

tft.drawFastHLine(0, tft.height()/2, tft.width(), 0x8410);

tft.drawFastVLine(tft.width()/2, 0, tft.height(), 0x8410);

//LEVEL BAR

pinMode(LED\_BUILTIN, OUTPUT);

Serial.begin(115200);

while (!Serial){}

Wire1.begin();

delay(100);

if (imu.begin(0x6B, 0x1E, Wire1)==false) {

while (1) {

digitalWrite(LED\_BUILTIN, HIGH);

delay(10);

digitalWrite(LED\_BUILTIN, LOW);

delay(20);

}

}

for (int i = 0; i < 5; i++) {

digitalWrite(LED\_BUILTIN, HIGH);

delay(100);

digitalWrite(LED\_BUILTIN, LOW);

delay(50);

}

}

void loop() {

if (imu.accelAvailable() && imu.magAvailable() && imu.gyroAvailable()) {

imu.readAccel();

imu.readMag();

imu.readGyro();

get\_IMU(Axyz, Mxyz, Gxyz);

Gxyz[0]= imu.calcGyro(imu.gx) \* DEG\_TO\_RAD;

Gxyz[1]= imu.calcGyro(imu.gy) \* DEG\_TO\_RAD;

Gxyz[2]= imu.calcGyro(imu.gz) \* DEG\_TO\_RAD;

Axyz[0] = imu.calcAccel(Axyz[0]) \* GRAVITY;

Axyz[1] = imu.calcAccel(Axyz[1]) \* GRAVITY;

Axyz[2] = imu.calcAccel(Axyz[2]) \* GRAVITY;

Mxyz[0] = imu.calcMag(Mxyz[0]);

Mxyz[1] = imu.calcMag(Mxyz[1]);

Mxyz[2] = imu.calcMag(Mxyz[2]);

deltat = filter.deltatUpdate();

filter.MadgwickUpdate( -Gxyz[0], Gxyz[1], Gxyz[2], // Flip Gyro Handedness

-Axyz[0], Axyz[1], Axyz[2], // Flip Accel Handedness

Mxyz[0], Mxyz[1], Mxyz[2], deltat);

/\*filter.MahonyUpdate( -Gxyz[0], Gxyz[1], Gxyz[2], // Flip Gyro Handedness

-Axyz[0], Axyz[1], Axyz[2], // Flip Accel Handedness

Mxyz[0], Mxyz[1], Mxyz[2], deltat);\*/

if (millis() - lastPrint >= PRINT\_INTERVAL) {

lastPrint = millis();

yaw = (int)filter.getYaw() - 180; // yaw range = [-180, 180]

pitch = (int)filter.getPitch();

roll = (int)filter.getRoll();

Serial.print("Orientation: ");

Serial.print(yaw);

Serial.print(", ");

Serial.print(pitch);

Serial.print(", ");

Serial.println(roll);

if(yaw >= 80) // out of range at +/- 80 degrees

yaw = 80;

else if(yaw <= -80)

yaw = -80;

twoDots(yaw, 120, 0, 100); //too slow, displays both dots

//==================================================

// Debugging code

//==================================================

/\*

Serial.print("A: ");

Serial.print(-Axyz[0]);

Serial.print(", ");

Serial.print(Axyz[1]);

Serial.print(", ");

Serial.println(Axyz[2]);

Serial.print("G: ");

Serial.print(-Gxyz[0]);

Serial.print(", ");

Serial.print(Gxyz[1]);

Serial.print(", ");

Serial.println(Gxyz[2]);

Serial.print("M: ");

Serial.print(Mxyz[0]);

Serial.print(", ");

Serial.print(Mxyz[1]);

Serial.print(", ");

Serial.println(Mxyz[2]);

float \* q;

q = filter.getQuat();

Serial.print("Quaternion: ");

Serial.print(q[0], 4);

Serial.print(", ");

Serial.print(q[1], 4);

Serial.print(", ");

Serial.print(q[2], 4);

Serial.print(", ");

Serial.println(q[3], 4);

\*/

}

}

}

//==================================================

// Sensor calibration

//==================================================

// Gyroscope scale 6194.6 to normalize

float G\_B[3] {220.78, -262.23, 254.91};

float G\_Ainv[3][3] {{ 0.88241, -0.01470, -0.00232},

{ -0.01470, 1.03994, 0.00201},

{ -0.00232, 0.00201, 1.08761}};

// Accelerometer scale 16477.7 to normalize

float A\_B[3] {-247.51, -81.74, 316.41};

float A\_Ainv[3][3] {{ 1.00076, 0.00025, -0.00001},

{ 0.00025, 1.00251, -0.00667},

{ -0.00001, -0.00667, 1.00254}};

// Magnetometer scale 3936.7 to normalize

float M\_B[3] {21.74, -129.71, -3045.94};

float M\_Ainv[3][3] {{ 1.48159, 0.05738, -0.00065},

{ 0.05738, 1.48909, -0.00508},

{ -0.00065, -0.00508, 1.53282}};

//==================================================

// Apply calibration bias offset and scale factors

//==================================================

void get\_IMU(float Axyz[3], float Mxyz[3], float Gxyz[3]) {

byte i;

float temp[3];

Axyz[0] = imu.ax;

Axyz[1] = imu.ay;

Axyz[2] = imu.az;

Mxyz[0] = imu.mx;

Mxyz[1] = imu.my;

Mxyz[2] = imu.mz;

Gxyz[0] = imu.gx;

Gxyz[1] = imu.gy;

Gxyz[2] = imu.gz;

// Gyroscope

for (i = 0; i < 3; i++) temp[i] = Gxyz[i] - G\_B[i];

Gxyz[0] = G\_Ainv[0][0] \* temp[0] + G\_Ainv[0][1] \* temp[1] + G\_Ainv[0][2] \* temp[2];

Gxyz[1] = G\_Ainv[1][0] \* temp[0] + G\_Ainv[1][1] \* temp[1] + G\_Ainv[1][2] \* temp[2];

Gxyz[2] = G\_Ainv[2][0] \* temp[0] + G\_Ainv[2][1] \* temp[1] + G\_Ainv[2][2] \* temp[2];

// Accelerometer

for (i = 0; i < 3; i++) temp[i] = Axyz[i] - A\_B[i];

Axyz[0] = A\_Ainv[0][0] \* temp[0] + A\_Ainv[0][1] \* temp[1] + A\_Ainv[0][2] \* temp[2];

Axyz[1] = A\_Ainv[1][0] \* temp[0] + A\_Ainv[1][1] \* temp[1] + A\_Ainv[1][2] \* temp[2];

Axyz[2] = A\_Ainv[2][0] \* temp[0] + A\_Ainv[2][1] \* temp[1] + A\_Ainv[2][2] \* temp[2];

// Magnetometer

for (i = 0; i < 3; i++) temp[i] = Mxyz[i] - M\_B[i];

Mxyz[0] = M\_Ainv[0][0] \* temp[0] + M\_Ainv[0][1] \* temp[1] + M\_Ainv[0][2] \* temp[2];

Mxyz[1] = M\_Ainv[1][0] \* temp[0] + M\_Ainv[1][1] \* temp[1] + M\_Ainv[1][2] \* temp[2];

Mxyz[2] = M\_Ainv[2][0] \* temp[0] + M\_Ainv[2][1] \* temp[1] + M\_Ainv[2][2] \* temp[2];

}