#include <SPI.h>

#include <Pixy2.h>

#define lh\_dir 8

#define lh\_pwm 2

#define lh\_enc 20

#define lv\_dir 10

#define lv\_pwm 4

#define lv\_enc 19

#define rv\_dir 11

#define rv\_pwm 5

#define rv\_enc 18

#define rh\_dir 9

#define rh\_pwm 3

#define rh\_enc 21

#define links 44

#define mitte 45

#define rechts 46

#define TOF\_Eingang 23

//ENCODER

volatile long ENCODER\_value\_LH = 0;

volatile long ENCODER\_value\_LV = 0;

volatile long ENCODER\_value\_RV = 0;

volatile long ENCODER\_value\_RH = 0;

Pixy2 pixy;

void setup() {

//Definieren der Pins

pinMode(lh\_pwm, OUTPUT);

pinMode(lh\_dir, OUTPUT);

pinMode(lv\_pwm, OUTPUT);

pinMode(lv\_dir, OUTPUT);

pinMode(rh\_pwm, OUTPUT);

pinMode(rh\_dir, OUTPUT);

pinMode(rv\_pwm, OUTPUT);

pinMode(rv\_dir, OUTPUT);

pinMode(rechts, OUTPUT);

pinMode(links, OUTPUT);

pinMode(mitte, OUTPUT);

pinMode(lh\_enc, INPUT\_PULLUP);

pinMode(lv\_enc, INPUT\_PULLUP);

pinMode(rh\_enc, INPUT\_PULLUP);

pinMode(rv\_enc, INPUT\_PULLUP);

pinMode(TOF\_Eingang, INPUT\_PULLUP);

//Geschwindigkeit auf Null setzen

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

digitalWrite(lh\_enc, HIGH);

digitalWrite(lv\_enc, HIGH);

digitalWrite(rh\_enc, HIGH);

digitalWrite(rv\_enc, HIGH);

attachInterrupt(digitalPinToInterrupt(lh\_enc), ENCODER\_LH, RISING);

attachInterrupt(digitalPinToInterrupt(lv\_enc), ENCODER\_LV, RISING);

attachInterrupt(digitalPinToInterrupt(rh\_enc), ENCODER\_RH, RISING);

attachInterrupt(digitalPinToInterrupt(rv\_enc), ENCODER\_RV, RISING);

Serial.begin(9600);

pixy.init();

}

void loop() {

//290 steps sind eine 90 Grad Drehung

// delay(10000);

// MOVE\_forward(100, 800);

delay(5000);

/\*

MOVE\_forward(100, 680);

TURN\_right(100, 290);

MOVE\_forward(100, 350);

MOVE\_rightforward(100, 800);

delay(5000);

MOVE\_leftbackward(100, 800);

MOVE\_backward(100, 350);

TURN\_left(100, 290);

MOVE\_backward(100, 800);

\*/

/\*

MOVE\_forward(100, 300);

MOVE\_rightforward(100, 800);

TURN\_left(100, 145);

\*/

Serial.println("1");

MOVE\_PIXY(25);

Serial.println("2");

delay(1000000000000);

// MOVE\_forward(50, 500);

// delay(1000);

// MOVE\_PIXY(25);

// MOVE\_leftforward(100, 500);

// delay(1000);

// MOVE\_right(100, 290);

// delay(1000);

// MOVE\_forward(100, 500);

// delay(1000000);

// MOVE\_forward(100, 500);

// TURN\_left(100, 500);

// delay(1000);

}

void MOVE\_PIXY(int sp) {

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, HIGH);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, HIGH);

// Serial.println("Pixy wird aufgerufen");

while (digitalRead(TOF\_Eingang) == LOW) {

Serial.println("Test");

pixy.ccc.getBlocks();

delay(15);

if (pixy.ccc.numBlocks) {

if (pixy.ccc.blocks[0].m\_x < 140) {

//nach rechts

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, LOW);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, LOW);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

} else if (pixy.ccc.blocks[0].m\_x > 180) {

//nach Links

digitalWrite(lh\_dir, LOW);

digitalWrite(lv\_dir, HIGH);

digitalWrite(rv\_dir, LOW);

digitalWrite(rh\_dir, HIGH);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

} else if (pixy.ccc.blocks[0].m\_x >= 140 && pixy.ccc.blocks[0].m\_x <= 180) {

// GERADEAUS

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, HIGH);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, HIGH);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

}

}

}

}

void MOVE\_forward(int sp, int steps) {

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, HIGH);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, HIGH);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LH < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void MOVE\_backward(int sp, int steps) {

digitalWrite(lh\_dir, LOW);

digitalWrite(lv\_dir, LOW);

digitalWrite(rv\_dir, LOW);

digitalWrite(rh\_dir, LOW);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LH < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void MOVE\_left(int sp, int steps) {

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, LOW);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, LOW);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LH < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void MOVE\_right(int sp, int steps) {

digitalWrite(lh\_dir, LOW);

digitalWrite(lv\_dir, HIGH);

digitalWrite(rv\_dir, LOW);

digitalWrite(rh\_dir, HIGH);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LH < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void MOVE\_leftforward(int sp, int steps) {

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, LOW);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, LOW);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, 0);

while (ENCODER\_value\_LH < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void MOVE\_rightforward(int sp, int steps) {

digitalWrite(lh\_dir, LOW);

digitalWrite(lv\_dir, HIGH);

digitalWrite(rv\_dir, LOW);

digitalWrite(rh\_dir, HIGH);

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LV < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void MOVE\_leftbackward(int sp, int steps) {

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, LOW);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, LOW);

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LV < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void TURN\_left(int sp, int steps) {

digitalWrite(lh\_dir, LOW);

digitalWrite(lv\_dir, LOW);

digitalWrite(rv\_dir, HIGH);

digitalWrite(rh\_dir, HIGH);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LH < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void TURN\_right(int sp, int steps) {

digitalWrite(lh\_dir, HIGH);

digitalWrite(lv\_dir, HIGH);

digitalWrite(rv\_dir, LOW);

digitalWrite(rh\_dir, LOW);

analogWrite(lh\_pwm, sp);

analogWrite(lv\_pwm, sp);

analogWrite(rv\_pwm, sp);

analogWrite(rh\_pwm, sp);

while (ENCODER\_value\_LH < steps) {

}

analogWrite(lh\_pwm, 0);

analogWrite(lv\_pwm, 0);

analogWrite(rv\_pwm, 0);

analogWrite(rh\_pwm, 0);

CLEAR\_VALUE();

}

void ENCODER\_LH() {

ENCODER\_value\_LH++;

}

void ENCODER\_LV() {

ENCODER\_value\_LV++;

}

void ENCODER\_RH() {

ENCODER\_value\_RH++;

}

void ENCODER\_RV() {

ENCODER\_value\_RV++;

}

void CLEAR\_VALUE() {

ENCODER\_value\_LH = 0;

ENCODER\_value\_LV = 0;

ENCODER\_value\_RH = 0;

ENCODER\_value\_RV = 0;

}