```
/*
 *  Ex_12 - PID Controller
 *  Line Following Robot using PID Controller
 *
 *  PortExpander
 *  -- Bit 4 - 0
 *  -- Line sensor Module
 *  -- Bit 7 - 5
 *  -- End Stop switch Module
 *
 *  NeoPixel
 *  -- IO5
 */

#include <Wire.h>

#define I2C_SDA 19
#define I2C_SCL 18

#define PEXP_I2CADDR 0x23
#define OLED_I2CAADR 0x3C

#define NEO_PIXEL 5
#define LED_COUNT 3


// i2c Slave Co-processor - On the Core-Module
#define I2CADDR 0x13
// i2c Slave Co-processor - On the Robot MainBoard
#define I2CADDR_B 0x12

uint8_t expanderData;

byte attinySlaveArrayBoard[3];
```

```
//Write a byte to the IO expander
void IOexpanderWrite(byte address, byte _data) {
    Wire.beginTransmission(address);
    Wire.write(_data);
    Wire.endTransmission();
}

//Read a byte from the IO expander
uint8_t IOexpanderRead(int address) {
    uint8_t _data;
    Wire.requestFrom(address, 1);
    if(Wire.available()) {
        _data = Wire.read();
    }
    return _data;
}



// Control the LED state of the LED on the
// Robotic MainBoard.
// Input:
//        outputState - HIGH/LOW
// Return:
//        0 - Sucess
//        1 - i2C write failure

int setBoardLED(uint8_t outputState) {
    attinySlaveArrayBoard[0] = 0x02;  // Command 0x02
    attinySlaveArrayBoard[1] = outputState? 0x01:0x00;  // Param1 -
LED State
    attinySlaveArrayBoard[2] = 0x00;  // Param2 - Dummy in this case
    delay(10);
    Wire.beginTransmission(I2CADDR_B);
    Wire.write(attinySlaveArrayBoard, 3); // Sends 3 bytes i2c to
Co-processor.
    if (Wire.endTransmission () == 0) { // Receive 0 = success (ACK
```

```
response)
        Serial.println("i2c Write to 0x12 Sucessfull");
        return 0;
    }
    else {
        Serial.println("i2c Write Failed");
        return 1;
    }
}

// Control the Stepper on the
// Robotic MainBoard.
// Input:
//        motorState - HIGH/LOW
// Return:
//        0 - Sucess
//        1 - i2C write failure

int setMotorRunning(uint8_t motorState) {
    attinySlaveArrayBoard[0] = 0x01;  // Command 0x01
    attinySlaveArrayBoard[1] = motorState? 0x01:0x00;  // Param1 -
Stop/Run
    attinySlaveArrayBoard[2] = 0x00;  // Param2 - Dummy in this case
    delay(10);
    Wire.beginTransmission(I2CADDR_B);
    Wire.write(attinySlaveArrayBoard, 3); // Sends 3 bytes i2c to
Co-processor.
    if (Wire.endTransmission () == 0) { // Receive 0 = success (ACK
response)
        Serial.println("i2c Write to 0x12 Sucessfull");
        return 0;
    }
    else {
        Serial.println("i2c Write Failed");
        return 1;
    }
}
```

```
// Control the Stepper on the
// Robotic MainBoard.
// Input:
//        motor - 0 (Motor A)
//                1 (Motor B)
//        direction - 0 -> Stop
//                    1 -> Clockwise
//                    2 -> Counter-Clockwise
// Return:
//        0 - Sucess
//        1 - i2C write failure

#define STOP 0
#define CW 1
#define CCW 2

int setDirection(int motor, byte direction) {
    attinySlaveArrayBoard[0] = motor == 0 ? 0x13 : 0x23;  //
Command 0x13 or 0x23
    attinySlaveArrayBoard[1] = (direction >= 0) && (direction <= 2)
? direction: 0;
                                                          // Param1
- Stop/CW/CCW
    attinySlaveArrayBoard[2] = 0x00;  // Param2 - Dummy in this case
    delay(10);
    Wire.beginTransmission(I2CADDR_B);
    Wire.write(attinySlaveArrayBoard, 3); // Sends 3 bytes i2c to
Co-processor.
    if (Wire.endTransmission () == 0) { // Receive 0 = success (ACK
response)
        Serial.println("i2c Write to 0x12 Sucessfull");
        return 0;
    }
    else {
        Serial.println("i2c Write Failed");
        return 1;
```

```
    }
}


// Control the Stepper on the
// Robotic MainBoard.
// Input:
//        motor - 0 (Motor A)
//                1 (Motor B)
//        rpm(Speed) - rpm*100 (L)
//        rpm(Speed) - rpm*100 (H)
//
//
// Return:
//        0 - Sucess
//        1 - i2C write failure


int setRPM(int motor, float rpm) {

    unsigned int rpm_x_100 = (int) (rpm * 100);

    attinySlaveArrayBoard[0] = motor == 0 ? 0x14 : 0x24;  //
Command 0x14 or 0x24
    attinySlaveArrayBoard[1] = (rpm_x_100 & 0xff);        // Param1
- rpm*100 (L)
    attinySlaveArrayBoard[2] = (rpm_x_100 >> 8) & 0xff;   // Param2
- Param1 - rpm*100 (H)
    delay(10);
    Wire.beginTransmission(I2CADDR_B);
    Wire.write(attinySlaveArrayBoard, 3); // Sends 3 bytes i2c to
Co-processor.
    if (Wire.endTransmission () == 0) { // Receive 0 = success (ACK
response)
        Serial.println("i2c Write to 0x12 Sucessfull");
        return 0;
    }
```

```
    else {
        Serial.println("i2c Write Failed");
        return 1;
    }
}

int8_t pidErrorMap[9][6] =
{
    {1, 1, 1, 1, 0, 4},
    {1, 1, 1, 0, 0, 3},
    {1, 1, 1, 0, 1, 2},
    {1, 1, 0, 0, 1, 1},
    {1, 1, 0, 1, 1, 0},
    {1, 0, 0, 1, 1, -1},
    {1, 0, 1, 1, 1, -2},
    {0, 0, 1, 1, 1, -3},
    {0, 1, 1, 1, 1, -4},
};

float Kp=4.3, Ki=0.0008, Kd=0;
const float minSpeed = 0.5;
const float maxSpeed = 10;

float error=0, P=0, I=0, D=0, PID_value=0;
float previousError=0;
uint8_t sensor[5] = {0, 0, 0, 0, 0};

void readSensorValues()
{
    expanderData = IOexpanderRead(PEXP_I2CADDR);
    sensor[0] = bitRead(expanderData, 0);
    sensor[1] = bitRead(expanderData, 1);
    sensor[2] = bitRead(expanderData, 2);
    sensor[3] = bitRead(expanderData, 3);
    sensor[4] = bitRead(expanderData, 4);
    for (byte i = 0; i < 9; i++) {
        if (sensor[0] == pidErrorMap[i][0] && sensor[1] ==
```

```
pidErrorMap[i][1] &&
            sensor[2] == pidErrorMap[i][2] && sensor[3] ==
pidErrorMap[i][3] &&
            sensor[4] == pidErrorMap[i][4]) {
            error = pidErrorMap[i][5];
        }
        if (sensor[0] + sensor[1] + sensor[2] + sensor[3] +
sensor[4] == 5) {
            // No Line??
        }
        else if (sensor[0] + sensor[1] + sensor[2] + sensor[3] +
sensor[4] == 0) {
            // Full Line??
        }
        else {
            // lastDefinedError = error;
        }
    }
}

void calculatePID()
{
    P = error;
    I = I + error;
    D = error - previousError;
    PID_value = (Kp*P) + (Ki*I) + (Kd*D);
    previousError = error;
}

void setMotorSpeed(int left, int right)
{
    // Sorry kids...
    // It is Left viewing from the front of the robot
    // And not the from the drivers seat. :(

    Serial.print("Left = "); Serial.print(String(left));
    Serial.print(" Right = ");Serial.println(String(right));
```

```
    delay(100);

    setRPM(0, left);
    setRPM(1, right);
}

void motorControl()
{
    float leftMotorSpeed = maxSpeed - PID_value;
    float rightMotorSpeed = maxSpeed + PID_value;

    leftMotorSpeed = constrain(leftMotorSpeed, minSpeed, maxSpeed);
    rightMotorSpeed = constrain(rightMotorSpeed, minSpeed,
maxSpeed);

    setMotorSpeed(leftMotorSpeed, rightMotorSpeed);
}


void setup() {
    Wire.begin(I2C_SDA, I2C_SCL);
    Serial.begin(115200); //set up serial library baud rate to
115200

    // Initialize PCF8574 configuration -> Pull High to Read
    IOexpanderWrite(PEXP_I2CADDR, 0xff);

    delay(2000);

    setDirection(0, CCW);
    setDirection(1, CW);
    setMotorRunning(HIGH);
}

void loop() {
    readSensorValues();
    calculatePID();
```

```
    motorControl();
}
```