ProgSeq

Generated on Sat Feb 26 2022 11:41:12 for ProgSeq by Doxygen 1.9.3

Sat Feb 26 2022 11:41:12

1 Class Documentation

1.1 progSeq Class Reference

Public Member Functions

- void **waitForButton** ()

  Blocking function that waits for the joystick button to be pressed.
- void **confirmCalibration** ()

  Blocking function that waits for the joystick button to be pressed, while displaying the black line position to check if the calibration has been done properly.
- void screen (String text)

  Display a text on the oled screen.
- void **calibrate** ()

  Start a calibration (the robots turns left then right to find the highest and lowest possible brightness for the floor).
- void setSpeed (int left, int right)

  Set the speed for left and right motors.
- void followLine (int maxSpeed)

  Execute a follow line code.
- void readSensors ()

  Read the line sensors value and store them in RAM for further readings.
- int getSensor (int index)

  Get a specific line sensor value.
- int getDistance ()

  Get the ultrasonic range distance.
- void readObstacle ()

  Read the front infrared proximity sensors values and store them in RAM for further readings.
- bool getObstacle (byte sensor)

  Get a specific infrared proximity sensor value.
- void setColor (int i, uint32_t color)

  Set the RGB LED i to the color of your choice.
- void **beepOn** ()

  Start the buzzer.
- void **beepOff** ()

  Stop the buzzer.
- int getJoystick ()

  Get the Joystick position.

1.1.1 Member Function Documentation

1.1.1.1 followLine() `void progSeq::followLine ( int maxSpeed )`

Execute a follow line code.

When this is executed in a loop, the robot will read the line position and modify its motor speeds to go forward while saying on the line.

Parameters

maxSpeed	The running speed of the robot (0 to 200 recommended)

1.1.1.2 getDistance() `int progSeq::getDistance ( )`

Get the ultrasonic range distance.

Returns

Distance in cm.

1.1.1.3 getJoystick() `int progSeq::getJoystick ( )`

Get the Joystick position.

Returns

`JOY_UNKNOWN` if the position is unknown.

'JOY_CENTER', `JOY_LEFT`, `JOY_RIGHT`, `JOY_UP` or `JOY_DOWN` in other cases.

1.1.1.4 getObstacle() `bool progSeq::getObstacle ( byte sensor )`

Get a specific infrared proximity sensor value.

NOTE : To get updated values, you must first run readObstacle().

Parameters

sensor	The sensor you want to read value from (`LEFT` or `RIGHT`).

Returns

true if there is an obstacle.

false if there is no obstacle.

1.1.1.5 getSensor() `int progSeq::getSensor ( int index )`

Get a specific line sensor value.

NOTE : To get updated values, you must first run readSensors().

Parameters

index	The id of the sensor tou want to get the value from, between 0 and 5.

Returns

The value of the sensor, between 0 and 1000;.

1.1.1.6 readObstacle() `void progSeq::readObstacle ( )`

Read the front infrared proximity sensors values and store them in RAM for further readings.

NOTE : As it's a void, it doesn't return anything, you have to call getObstacle(byte sensor) to get the actual values for each of the 2 sensors.

1.1.1.7 readSensors() `void progSeq::readSensors ( )`

Read the line sensors value and store them in RAM for further readings.

NOTE : As it's a void, it doesn't return anything, you have to call getSensor(int index) to get the actual values for each of the 5 sensors.

1.1.1.8 screen() `void progSeq::screen ( String text )`

Display a text on the oled screen.

Parameters

text	Multiline text to display, add a `\n` to print on the next line.

1.1.1.9 setColor() `void progSeq::setColor (`

```
int i,
uint32_t color )
```

Set the RGB LED i to the color of your choice.

Parameters

i	The index of the led you want to set the color.
color	An RGB color, you can use `RED`, `BLUE`, `GREEN`, `BLACK` and `WHITE`, but you can also use hexadecimal codes (`0xRRGGBB`).

1.1.1.10 setSpeed() `void progSeq::setSpeed (`

```
int left,
int right )
```

Set the speed for left and right motors.

Parameters

left	An integer between -255 (backwards full speed) and 255 (forward fulls speed) for the left motor
right	An integer between -255 (backwards full speed) and 255 (forward fulls speed) for the right motor

The documentation for this class was generated from the following files:

- progSeq.h
- progSeq.cpp

2 File Documentation

2.1 progSeq.h

```
1 #ifndef PROGSEQ_H
2 #define PROGSEQ_H
3
4 #include "Adafruit_GFX.h"
5 #include <Adafruit_NeoPixel.h>
6 #include <Adafruit_SSD1306.h>
7 #include <TRSensors.h>
8 #include <Wire.h>
9
10 #include "Arduino.h"
11
12 #define PWMA 6   // Left Motor Speed pin (ENA)
13 #define AIN2 A0  // Motor-L forward (IN2).
14 #define AIN1 A1  // Motor-L backward (IN1)
15 #define PWMB 5   // Right Motor Speed pin (ENB)
16 #define BIN1 A2  // Motor-R forward (IN3)
17 #define BIN2 A3  // Motor-R backward (IN4)
18 #define PIN 7
19 #define NUM_SENSORS 5
20 #define OLED_RESET 9
21 #define OLED_SA0 8
22 #define Addr 0x20
23 #define IR 4  // he infrare remote receiver pin
24 #define ECHO 2
25 #define TRIG 3
```

```
26
27 #define KEY2 0x18        // Key:2
28 #define KEY8 0x52        // Key:8
29 #define KEY4 0x08        // Key:4
30 #define KEY6 0x5A        // Key:6
31 #define KEY1 0x0C        // Key:1
32 #define KEY3 0x5E        // Key:3
33 #define KEY5 0x1C        // Key:5
34 #define SpeedDown 0x07   // Key:VOL-
35 #define SpeedUp 0x15     // Key:VOL+
36 #define ResetSpeed 0x09  // Key:EQ
37 #define Repeat 0xFF      // press and hold the key
38
39 #define BLACK 0x000000
40 #define RED   0xFF0000
41 #define GREEN 0x00FF00
42 #define BLUE  0x0000FF
43 #define WHITE 0xFFFFFF
44
45 #define RIGHT 0X0F
46 #define LEFT  0xF0
47
48 #define JOY_UNKNOWN -1
49 #define JOY_LEFT 1
50 #define JOY_RIGHT 2
51 #define JOY_UP 3
52 #define JOY_DOWN 4
53 #define JOY_CENTER 5
54
55
56 #define beep_on PCF8574Write(0xDF & PCF8574Read())
57 #define beep_off PCF8574Write(0x20 | PCF8574Read())
58
59 extern Adafruit_SSD1306 display;
60
61 extern Adafruit_NeoPixel RGB;
62
63 class progSeq {
64  public:
65   progSeq();
66
71   void waitForButton();
76   void confirmCalibration();
82   void screen(String text);
87   void calibrate();  // calibrate line sensors
94   void setSpeed(int left, int right);  // -255 (backward) to 255 (forward)
101   void followLine(int maxSpeed);  // move motors according to line position
106   void readSensors();  // update sensors state
114   int getSensor(int index);  // get sensors values, index 0 to 5
120   int getDistance();
126   void readObstacle();
135   bool getObstacle(byte sensor);
142   void setColor(int i, uint32_t color);
147   void beepOn();
152   void beepOff();
159   int getJoystick();
160
161  private:
162   unsigned int sensorValues[NUM_SENSORS];
163
164   TRSensors trs = TRSensors();
165
166   int obstacle;
167
168   void PCF8574Write(byte data);
169   byte PCF8574Read();
170 };
171
172 #endif
173
174
```

Index