Digilent RDKcK Line Sensor Demo Project for chipKIT™ MPIDE

Revision: May 16, 2012

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

This demo project for the RDKcK Line Sensor was written with MPIDE 0023. It is meant to show the basic functionality of the RDKcK Line Sensor, as well as basic programming instructions using chipKIT™ MPIDE. This document provides an overview of the demo project for the RDKcK Line Sensor using a Digilent MX3cK microcontroller board.

Reference manuals for the MX3cK microcontroller and the accompanying Pmods can be found on Digilent's website.

Overview

Please refer to the Line Follower Assembly document link for step by step instructions to build the RDKcK Line Sensor robot. Note that lithium based AA batteries will be best for powering the robot.

For this demo, the Infrared (IR) proximity sensors are used to sense a line composed of nonreflecting material will work best. Setting the centers of the two inner sensors, assuming a 4 sensor system, are set approximately 2 centimeters apart. All four sensors should be approximately 4 millimeters above the surface. The sensors will be most responsive if a matte black line is followed over a white background. The black line can be made with electric tape, or, dry erase marker. If dry erase marker is used, make sure the line is around ½" thick. This document assumes a line made of black electric tape with a white background.

How to upload the project on to the RDKcK Line following robot (for more detailed instructions see the assembly document):

1. Make sure that the downloaded demo files are saved in your sketches folder (Sketchbook).
2. Open MPIDE.
3. Navigate to File->Sketchbook->RDKcK Line Demo->RDKcK_Line_Demo
4. When the demo is open, navigate to Tools->Board and select MX3cK
5. Connect the MX3cK to the computer via the included Digilent USB A to mini B cable.
6. Navigate to Tools->Serial Port and select the port that the board is connected to.
7. Click the "Verify and Upload" button second button in from the top left of the MPIDE window.

Robot Actions Explained

The RDKcK Line Sensor robot will alter its direction based on the four IR sensors. When the surface below the IR sensor is dark, less light will be reflected back to the sensor and the pin that the sensor is connected to will output LOW or, logic level '0'. Table 1 shows all possible sensor outputs and the corresponding action that the robot will take.

Table 1. Robot Actions

instruction Value	Sensor Output FRS	MRS	MLS	FLS	Action Description
0	0	0	0	0	Robot stops
1	0	0	0	1	Turn Right
2	0	0	1	0	Continue
3	0	0	1	1	Turn Right
4	0	1	0	0	Continue
5	0	1	0	1	Continue
6	0	1	1	0	Continue
7	0	1	1	1	Turn Right
8	1	0	0	0	Turn Left
9	1	0	0	1	Continue
10	1	0	1	0	Continue
11	1	0	1	1	Turn Right
12	1	1	0	0	Turn Left
13	1	1	0	1	Turn Left
14	1	1	1	0	Turn Left
15	1	1	1	1	Move Forward

“Continue” means the robot will not change from the current operation.

Function Descriptions

void setup()
Parameters
- None.

Return value
- None.

This function must be included in all MPIDE sketches. This is the fundamental setup function used to initialize the board; this sets the starting state of the processor before the loop function runs. Within the setup function, all I/O pins (PmodLS1 pins and PmodCON3 pins) are set as either input or output with the built in pinMode() function.

void loop()
Parameters
- None.

Return value
- None.

This function must be included in all MPIDE sketches. This function is the processing section of the program all the steps you want the processor to go through will be defined here. The steps that occur in this function assume that everything is preconfigured in the setup function and is ready to run. In most sketches this function will make calls to other user defined functions to preform steps or tasks. In this demo in the loop function calls will be made to checkSensors(),stopRobot(), forward(), turnRight() and turnLeft(), all described later in this document. Based on what value checkSensors() returns, the

robot will then make calls to the other four functions described above to carry out an action based on Table 1 above. This is called “polling”.

int checkSensors()
Parameters
- None.

Return value
- int sensorConfig

This function reads the state of each sensor from the PmodLS1 and returns the value as an integer. The state of each sensor is read using the digitalRead() function on its respective pin. The state will either be HIGH (1) if the sensor is on (sees reflected light i.e. white background), or LOW (0) if the sensor is off (sees no light i.e. black background). Each state is multiplied by a hexadecimal byte value and then logically OR’d together with the other three sensor states to create the byte sensorConfig.

Based on Table 1 above, the state of FRS will be the most significant bit and the state of FLS will be the least significant bit. Therefore, the state of FRS will be multiplied by 0x08 (0b00001000), MRS will be multiplied by 0x04 (0b00000100), MLS will be multiplied by 0x02 (0b00000010), and FLS will be multiplied by 0x01 (0b00000001).

Below is an example of this process.

1. If FRS is on, then digitalRead() on that pin will return HIGH, or logic level ‘1’.
    a. byte sense1 = digitalRead(farRight)*(0x08);
        i. frs = 0x08 = 0b00001000
2. If MRS is off, then digitalRead() on that pin will return LOW, or logic level ‘0’.
    a. byte mrs = digitalRead(midRight)*(0x04);
        i. mrs = 0x00 = 0b00000000
3. If MLS is on, then digitalRead() on that pin will return HIGH, or logic level ‘1’.
    a. byte mls = digitalRead(midLeft)*(0x02);
        i. mls = 0x02 = 0b00000010
4. If FLS is off, then digitalRead() on that pin will return LOW, or logic level ‘0’.
    a. byte fls = digitalRead(farLeft)*(0x01);
        i. fls = 0x00 = 0b00000000
5. frs | mrs | mls | fls = 0x0A = 0b00001010
6. int(sensorConfig) returns 10.

The following functions are called based on the values that are returned from checkSensors(). These are the actions that are described in the beginning of the document and in table 1.

void stopRobot()
Parameters
- None.

Return value
- None.

This function stops the robot by setting the servo angle to 90, and detaching the servo pins.

void forward()
Parameters

- None.

Return value

- None.

This function will move the robot forward. When the program is uploaded to the board, it will move forward until one of the sensors triggers low.

void turnRight()

Parameters

- None.

Return value

- None.

This function will turn the robot to the right.

void turnLeft()

Parameters

- None.

Return value

- None.

This function will turn the robot to the left.