"Infrared Distance Sensor Array for Line-Following-Robot"

Introduction:

This lab will introduce you to IR Distance sensor array and will make you learn how to use it to for a following robot. We will make our own distance sensor array based on Pololu infrared transmitter and receiver, which can then be mounted on your robots. We will use the Pololu QTR distance array sensors library that will enable us to easily calibrate and configure the distance sensor array as a line follower sensor.

Pololu QTR sensor is a infrared LED (transmitter) and photo-transistor (receiver) pair that can be used as a distance sensor. We will make the analog distance sensors array using the infrared LED and transistor pair for line following. It is also available as a QTR-1A single sensor module and also as QTR-8A $8x$ sensor module. We will build our own QTR module with 6 sensors. Our module will consists of 6 units of the circuit shown in 1. The unit sensor has three input/output ports. The voltage pin, the ground pin, and the output pin. The diode and the transister in the circuit are the LED and photo-transistor pair. The Optimal sensing distance is 0.125" (3 mm), while the maximum recommended sensing distance is 0.25" (6 mm).

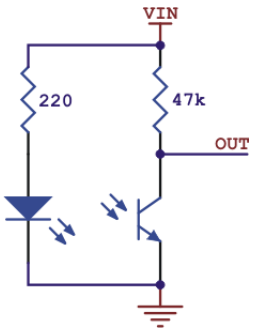

Figure 1: Unit Circuit for Distance Sensor copied from Reference[11]

Arduino library for Pololu QTR

The Pololu QTR also have a library that can be used with the sensor array. It helps us to configure, calibrate, and use to measure the distance. Download the archive from `https://github.com/pololu/qtr-sensors-arduino/archive/master.zip`, decompress it, and drag the "QTRSensors" folder to your arduino-1.0/ libraries directory. You should now be able to use these libraries in your sketches by selecting "Sketch" then, "Import Library" and then "QTRSensors" from your "Arduino IDE" to restart your Arduino IDE before it sees the new libraries. You can create a **QTRSensorsAnalog** object for your QTR-6A sensors. The first argument to the **QTRSensorsAnalog** constructor is an array of analog input pins (0 7) while the second is the number of sensors. The library take care of the configuration required to convert the raw sensor output to the usable distance sensor measurement.

```
#include <QTRSensors.h>

// create an object for QTR-6A sensors on analog inputs 0, 1, 2, 3, 4, and 5
QTRSensorsAnalog qtra((unsigned char[]) {0, 1, 2, 3 ,4, 5}, 6);
```

The library gives you a 3 ways to read the sensors.

- You can request raw sensor values using the read() method. May be used to check if the sensors are working properly and not damaged during soldering.
- You can request calibrated sensor values using the readCalibrated() method. Calibrated sensor values will always range from 0 to 1000, with 0 being as or more reflective (i.e. whiter) than the most reflective surface encountered during calibration, and 1000 being as or less reflective (i.e. blacker) than the least reflective surface encountered during calibration. you may use it to check if calibration is proper.
- For line-detection you can request the line location using the readLine() method. It provides calibrated values for each sensor and returns an integer that tells you where it thinks the line is. If you are using N sensors, a returned value of 0 means it thinks the line is on or to the outside of sensor 0, and a returned value of 1000 * (N-1) means it thinks the line is on or to the outside of sensor N-1. As you slide your sensors across the line, the line position will change monotonically from 0 to 1000 * (N-1), or vice versa.

The rest of the function provided by the library are defined at `http://www.pololu.com/docs/pdf/0J19/QTR_arduino_library.pdf`. It is also recommended to read the **QTRSensors.h** to understand the library functions provided by the library.

Instructions

- Implement the circuit shown in figure 1 on a proto-PCB.
- We will use to make 6 unit of sensors to make the QTR-6A module.
- Place the sensor on the PCB such that thet are equally spaced.
- Solder the PCB carefully, do not heatup the sensor too much otherwise it will be damaged.
- Once your QTR-6A is finished you can test it using the example code.

Sample - Code

```
#include <QTRSensors.h>

// This example is designed for use with six QTR-^A sensors
// These reflectance sensors should be connected to analog inputs 0 to 5.
// Change the EMITTER_PIN #define below from 2 to QTR_NO_EMITTER_PIN.

// The setup phase of this example calibrates the sensor for ten seconds and turns on
// the LED built in to the Arduino on pin 13 while calibration is going on.
// During this phase, you should expose each reflectance sensor to the lightest and
// darkest readings they will encounter.
// For example, if you are making a line follower, you should slide the sensors across the
// line during the calibration phase so that each sensor can get a reading of how dark the
// line is and how light the ground is.  Improper calibration will result in poor readings.
// If you want to skip the calibration phase, you can get the raw sensor readings
// (analog voltage readings from 0 to 1023) by calling qtra.read(sensorValues) instead of
// qtra.readLine(sensorValues).

// The main loop of the example reads the calibrated sensor values and uses them to
// estimate the position of a line.  You can test this by taping a piece of 3/4" black
// electrical tape to a piece of white paper and sliding the sensor across it.  It
// prints the sensor values to the serial monitor as numbers from 0 (maximum reflectance)
// to 1000 (minimum reflectance) followed by the estimated location of the line as a number
// from 0 to 5000.  1000 means the line is directly under sensor 1, 2000 means directly
// under sensor 2, etc.  0 means the line is directly under sensor 0 or was last seen by
```

```
// sensor 0 before being lost.  5000 means the line is directly under sensor 5 or was
// last seen by sensor 5 before being lost.


#define NUM_SENSORS             6  // number of sensors used
#define NUM_SAMPLES_PER_SENSOR  4  // average 4 analog samples per sensor reading
#define EMITTER_PIN             QTR_NO_EMITTER_PIN  // if emitter is not controlled by digital pin 2

// sensors 0 through 5 are connected to analog inputs 0 through 5, respectively
QTRSensorsAnalog qtra((unsigned char[]) {0, 1, 2, 3, 4, 5},
  NUM_SENSORS, NUM_SAMPLES_PER_SENSOR, EMITTER_PIN);
unsigned int sensorValues[NUM_SENSORS];


void setup()
{
  delay(500);
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);    // turn on Arduino's LED to indicate we are in calibration mode
  for (int i = 0; i < 400; i++)  // make the calibration take about 10 seconds
  {
    qtra.calibrate();       // reads all sensors 10 times at 2.5 ms per six sensors (i.e. ~25 ms per call)
  }
  digitalWrite(13, LOW);     // turn off Arduino's LED to indicate we are through with calibration

  // print the calibration minimum values measured when emitters were on
  Serial.begin(9600);
  for (int i = 0; i < NUM_SENSORS; i++)
  {
    Serial.print(qtra.calibratedMinimumOn[i]);
    Serial.print(' ');
  }
  Serial.println();

  // print the calibration maximum values measured when emitters were on
  for (int i = 0; i < NUM_SENSORS; i++)
  {
    Serial.print(qtra.calibratedMaximumOn[i]);
    Serial.print(' ');
  }
  Serial.println();
  Serial.println();
  delay(1000);
}


void loop()
{
  // read calibrated sensor values and obtain a measure of the line position from 0 to 5000
  // To get raw sensor values, call:
  //  qtra.read(sensorValues); //instead of unsigned int position = qtra.readLine(sensorValues);
  unsigned int position = qtra.readLine(sensorValues);

  // print the sensor values as numbers from 0 to 1000, where 0 means maximum reflectance and
  // 1000 means minimum reflectance, followed by the line position
  for (unsigned char i = 0; i < NUM_SENSORS; i++)
  {
    Serial.print(sensorValues[i]);
    Serial.print('\t');
```

```
  }
  //Serial.println(); // uncomment this line if you are using raw values
  Serial.println(position); // comment this line out if you are using raw values

  delay(250);
}
```

0.1 Notes for Line follower

We would like to make a line follower by using the sensor array. Therefore, during the calibration we need to calibrate the whole sensor array. You need to slide the array across the line during the calibration phase so the each sensor can get a reading of how dark the line is and how light the ground is. Make the calibration sheet by printing a Black dark line on a A4 size paper.

Although you have already seen a sample code, but for the autonomus line follower robot the routine might be modified slightly. It should look similar to the following.

```
#include <QTRSensors.h>
// create an object for your type of sensor (RC or Analog)
// in this example we have six sensors on analog inputs 0 - 5
QTRSensorsA qtr((char[]) {0, 1, 2, 3, 4, 5}, 6);
void setup()
{
// optional: wait for some input from the user, such as
a button press
// then start calibration phase and move the sensors over both
// reflectance extremes they will encounter in your application:
int i;
for (i = 0; i < 250; i++) // make the calibration take about 5 seconds
{
qtr.calibrate();
delay(20);
}
// optional: signal that the calibration phase is now over and wait for further
// input from the user, such as a button press

}
```

A sample routine for reading the sensors and line-following is given at `http://www.pololu.com/docs/pdf/0J19/QTR_arduino_library.pdf`. You can also try the the line follower with or without the PID controller.

References

1. `http://en.wikipedia.org/wiki/Proximity_sensor`.
2. `http://profesores.fi-b.unam.mx/m3615m/datasheet-sen136b5b.pdf`.
3. `http://www.seeedstudio.com/wiki/Ultra_Sonic_range_measurement_module`.
4. `http://arduino.cc`.
5. "Arduino Programming Notebook" by - by Brian W. Evans.
6. Beginning Android ADK with Arduino by Mario Bohmer.
7. `http://arduino.cc/en/uploads/Main/arduino_Uno_Rev3-02-TH.zip`.

8. http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf.
9. http://arduino.cc/en/Main/ArduinoBoardUno.
10. Computational Principles of Mobile Robotics by from Dudek and Jenkin.
11. http://www.pololu.com/docs/pdf/0J19/QTR_arduino_library.pdf.
12. https://github.com/pololu/qtr-sensors-arduino/archive/master.zip.
13. http://www.pololu.com/docs/0J19/all.
14. http://www.instructables.com/id/Arduino-Line-Following-Robot-for-Beginners/.