PAMANTASAN NG LUNGSOD NG MAYNILA
UNIVERSITY OF THE CITY OF MANILA
INTRAMUROS, MANILA

COLLEGE OF ENGINEERING AND TECHNOLOGY
COMPUTER ENGINEERING DEPARTMENT

CPE 423
MICROPROCESSOR SYSTEM

FINAL PROJECT
MICROCONTROLLER-BASED SMART PARKING SYSTEM

Members:	Student Number
DOMAEL, HEHERSON I.	2015-02185
FONTE, KARL VINCENT A.	2015-02216
GUERRA, BETHANY SHARMAINE R.	2015-20653
MUNGCAL, GIOMAR C.	2015-02364
PACHECO, REGINA JOYCE	2015-10467

Instructor:
ENGR. EVANGELINE P. LUBAO

Objectives

- To create a microcontroller-based prototype of a smart parking system, projected through a diorama of a small-scale parking space.
- To utilize the use of an LCD monitor display to indicate the number of occupied and vacant slots.
- To familiarize students with the functionality of a photoresistor and implement it in the prototype as a way to detect whether a parking spot is occupied.

Theory

Light-dependent Resistor (Photoresistor)

A photoresistor (or light-dependent resistor, LDR, or photo-conductive cell) is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits, and light-activated and dark-activated switching circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photoresistor can substantially differ among dissimilar devices. Moreover, unique photoresistors may react substantially differently to photons within certain wavelength bands.

How an LDR works

It is relatively easy to understand the basics of how an LDR works without delving into complicated explanations. It is first necessary to understand that an electrical current consists of the movement of electrons within a material. Good conductors have a large number of free electrons that can drift in a given direction under the action of a potential difference. Insulators with a high resistance have very few free electrons, and therefore it is hard to make the them move and hence a current to flow.

An LDR or photoresistor is made any semiconductor material with a high resistance. It has a high resistance because there are very few electrons that are free and able to move - the vast majority of the electrons are locked into the crystal lattice and unable to move. Therefore. in this state there is a high LDR resistance.

As light falls on the semiconductor, the light photons are absorbed by the semiconductor lattice and some of their energy is transferred to the electrons. This gives some of them sufficient energy to break free from the crystal lattice so that they can then conduct electricity. This results in a lowering of the resistance of the semiconductor and hence the overall LDR resistance.

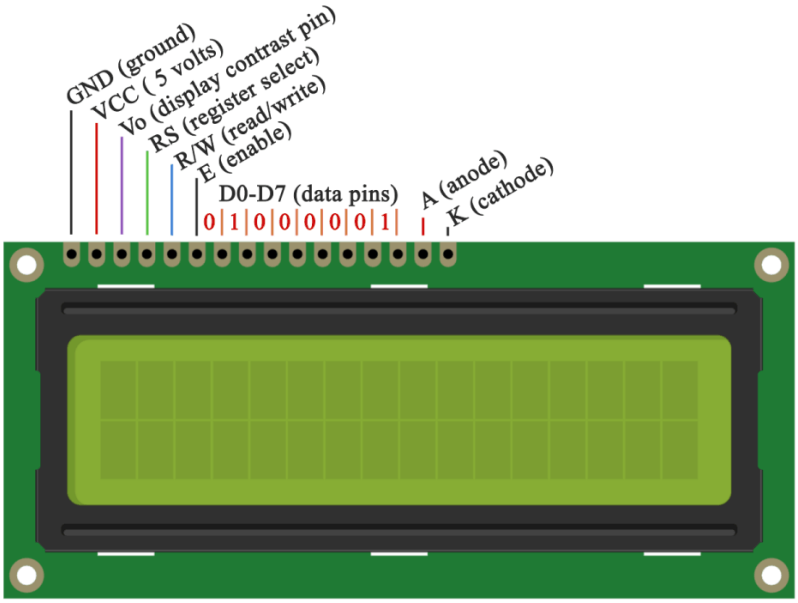
The process is progressive, and as more light shines on the LDR semiconductor, so more electrons are released to conduct electricity and the resistance falls further.

LCD Display

The LCD Pinout

It has 16 pins and the first one from left to right is the Ground pin. The second pin is the VCC which we connect the 5 volts pin on the Arduino Board. Next is the Vo pin on which we can attach a potentiometer for controlling the contrast of the display.

Next, The RS pin or register select pin is used for selecting whether we will send commands or data to the LCD. For example if the RS pin is set on low state or zero volts, then we are sending commands to the LCD like: set the cursor to a specific location, clear the display, turn off the display and so on. And when RS pin is set on High state or 5 volts we are sending data or characters to the LCD.



Next comes the R / W pin which selects the mode whether we will read or write to the LCD. Here the write mode is obvious and it is used for writing or sending commands and data to the LCD. The read mode is used by the LCD itself when executing the program which we don't have a need to discuss about it in this tutorial.

Next is the E pin which enables the writing to the registers, or the next 8 data pins from D0 to D7. Through these pins we are sending the 8 bits data when we are writing to the registers or for example if we want to see the latter uppercase A on the display we will send 0100 0001 to the registers according to the ASCII table.

And the last two pins A and K, or anode and cathode are for the LED back light. After all we don't have to worry much about how the LCD works, as the Liquid Crystal Library takes care for almost everything. From the Arduino's official website you can find and see the functions of the library which

enable easy use of the LCD. We can use the Library in 4 or 8 bit mode. In this tutorial we will use it in 4 bit mode, or we will just use 4 of the 8 data pins.

Servomotor

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback.

Because servo motors use feedback to determine the position of the shaft, you can control that position very precisely. As a result, servo motors are used to control the position of objects, rotate objects, move legs, arms or hands of robots, move sensors etc. with high precision. Servo motors are small in size, and because they have built-in circuitry to control their movement, they can be connected directly to an Arduino.

Most servo motors have the following three connections:

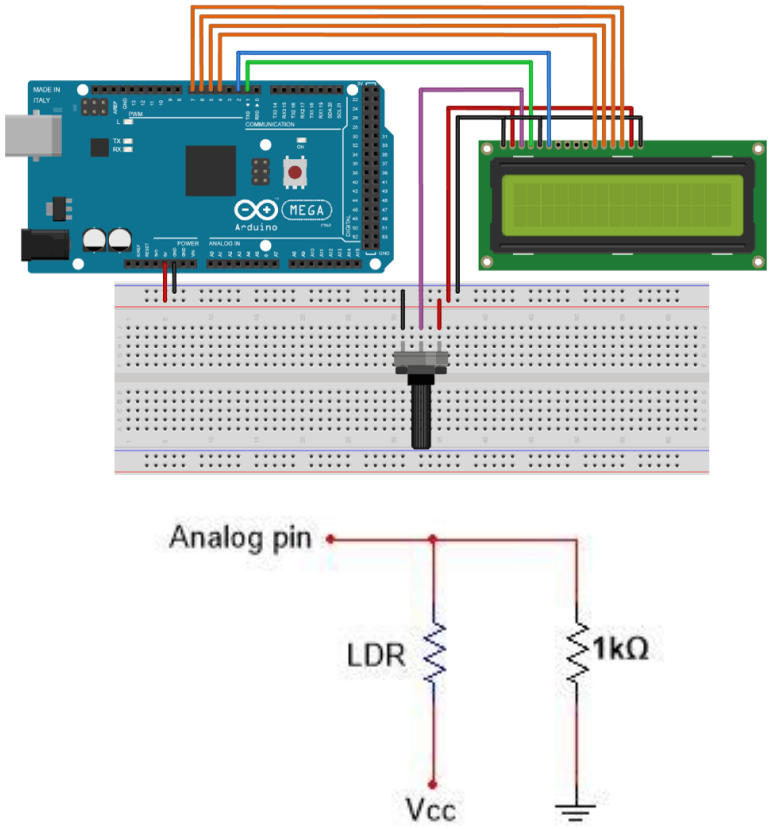
1. Black/Brown ground wire
2. Red power wire (around 5V)
3. Yellow or White PWM wire

Materials

QTY	Name of Material
1	Arduino MEGA 2560 R3
1	16x2 LCD Display
1	Digital Micro servo 9g SG92R
10	12 mm Light-dependent resistor
5	Male to Female Jump Wires
10	Mini Toy Cars
120	Male to Female Jumper Wires
40	Male to Male Jumper Wires

Procedures

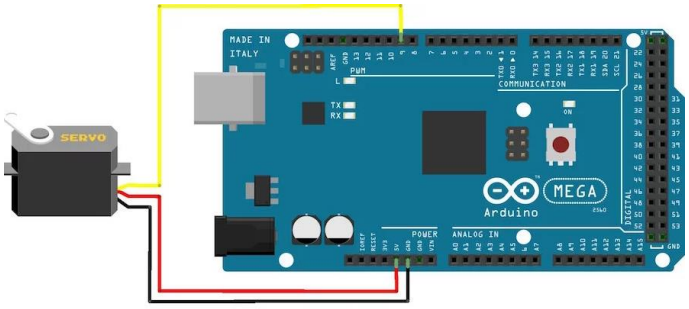
We will use just 6 digital input pins from the Arduino Board. The LCD's registers from D4 to D7 will be connected to Arduino's digital pins from 4 to 7. The Enable pin will be connected to pin number 2 and the RS pin will be connected to pin number 1. The R/W pin will be connected to Ground and the Vo pin will be connected to a 1K resistor instead of a potentiometer as shown on the circuit below.



Twelve pairs of LDR and resistor are connected to form a light dependent resistor for a specific slot. Ten pairs are used for parking slots, while two are used for entrance and exit slot. The pairs occupy twelve slots of the Arduino since it is necessary to determine the slots that are occupied and those that are not occupied.

Finally, the servo motor is connected as shown below. Controlling a servo motor directly from the Arduino is quite easy. However, a servo motor may require significantly more current than the Arduino can provide. The following example uses a standard sized servo (without any load) powered directly from the Arduino via USB. When powering the servo directly from the Arduino board:

1. Connect the black wire from the servo to the GND pin on the Arduino
2. Connect the red wire from the servo to the +5V pin on the Arduino
3. Connect the yellow or white wire from the servo to a digital pin on the Arduino



Integrate the LDRs, LCD, and servo motor into the microcontroller. Design the parking lot according to the proposed parking layout with ten slots.

Actual Experimentation

Conclusion

A smart parking system that determines the available and occupies slot numbers is hereby established. The status of each slot is determined in real-time via the LCD displayed outside the gate. Also, the gate automatically opens whenever a car occupies the LDR near the vicinity of the gate. Thus, this parking system can operate even without human intervention.

Recommendation

The following are recommended:

The parking lot may apply machine learning techniques so that the light dependent resistors may determine whether a car occupies the parking lot or not.

The parking system may use a mobile application updated in real-time so that parking patrons may determine the status of the parking slots even when they are not actually outside the gate.

References

- https://howtomechatronics.com/tutorials/arduino/lcd-tutorial/
- https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/
- https://en.wikipedia.org/wiki/Photoresistor
- https://www.electronics-notes.com/articles/electronic_components/resistors/light-dependent-resistor-ldr.php

Appendix

Arduino Code for Smart Parking System

```
#include <LiquidCrystal.h>
#include <Servo.h>

const int rs = 1, en = 2, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
Servo servo;

byte occupied[8] = {  B11111,
                      B11111,
                      B11111,
                      B11111,
                      B11111,
                      B11111,
                      B11111,
                      B11111  };
byte nonocc[8]={      B11111,
                      B10001,
                      B10001,
                      B10001,
                      B10001,
                      B10001,
                     B10001,
                    B11111  };

byte blank[8]={
  B00000,
  B00000,
  B00000,
  B00000,
  B00000,
  B00000,
  B00000,
  B00000
};

const int numSlots        = 10;
const int slot[numSlots]  = {A1, A2, A3, A4, A5, A6, A7, A8, A9,A10};
const int entGate         = A11;
const int exGate          = A0;

int numCarsIn = 0, angle = 5;
bool arrive = false, inSlot[10], slotted[10];
```

```
void setup()
{
  servo.attach(8);
  servo.write(5);
  lcd.begin(16,2);
  delay(1000);

  lcd.createChar(7,occupied);
  lcd.createChar(6,nonocc);
  lcd.createChar(5,blank);
  lcd.clear();
  for (int i=0; i<10; i++)
  {
    lcd.setCursor(i,0);
    if (i==9)
      lcd.print("A");
    else
      lcd.print(i+i);
  }
  lcd.setCursor(11,0);
  lcd.print("VIn: ");
  lcd.print(numCarsIn);
  for (int i=0; i<10; i++)
  {
    inSlot[i]   = false;
    lcd.setCursor(i,1);
    if (inSlot[i] == false)
      lcd.write(6);
    else
      lcd.write(7);
  }
}

void loop()
{
  lcd.setCursor(11,1);
  lcd.print("PLM50");

  lcd.setCursor(0,0);
  lcd.print("VehiIn: ");
  if (numCarsIn == 10)
    lcd.print("F");
  else
  lcd.print(numCarsIn);
```

```
  for (int i=0; i<10; i++)
  {
    lcd.setCursor(i,1);
    if (inSlot[i]==true)
      lcd.write(7);
    else
      lcd.write(6);
  }

  if (analogRead(entGate)<100 && numCarsIn<numSlots)    // A VEHICLE REQUESTS FOR
ENTRANCE
  {
    numCarsIn++;                              // NUMBER OF CARS INCREMENTED
    arrive = true;

    for (angle=0; angle<=90; angle++)              // GATE RISES SO VEHICLE MAY ENTER
    {
      servo.write(angle);
      delay(15);
    }
    delay(1500);
    for (angle=90; angle>0; angle--)
    {
      servo.write(angle);
      delay(15);
    }
    delay(50);
  }

if (numCarsIn!=0 && numCarsIn<=numSlots)
{
      for (int i=0; i<numSlots; i++)
      {
        if (i==0 || i==1 || i==7 || i==8)
        {
          if (analogRead(slot[i])<200 && inSlot[i]==false)
          {
            lcd.setCursor(i,1);
            lcd.write(7);
            inSlot[i] = true;
          }
          if (analogRead(slot[i]) >= 200 && inSlot[i]==true)
          {
            lcd.setCursor(i,1);
            lcd.write(6);
            inSlot[i] = false;
```

```
        }
       }
       else
       {
        if (analogRead(slot[i])<100 && inSlot[i]==false)
        {
           lcd.setCursor(i,1);
           lcd.write(7);
           inSlot[i] = true;
        }
        if (analogRead(slot[i]) >= 100 && inSlot[i]==true)
        {
           lcd.setCursor(i,1);
           lcd.write(6);
          inSlot[i] = false;
        }
       }
  }
}

  if (analogRead(exGate)<100 && numCarsIn>0) // A VEHICLE REQUESTS FOR EXIT
  {
    numCarsIn--;
    delay(500);
    for (angle=0; angle<=90; angle++)
    {
      servo.write(angle);
      delay(15);
    }
    delay(1500);
    for (angle=90; angle>0; angle--)
    {
      servo.write(angle);
      delay(15);
    }
    delay(50);
  }

  arrive = false;
}
```