Design and Implementation of Arduino Microcontroller Based Automatic Lighting Control with I2C LCD Display

Akinwole OO, Oladimeji TT

*Department of Electrical/Electronic Engineering, the Federal Polytechnic, Ado-Ekiti, Nigeria

Abstract

The project uses closed loop control system to automatically operate a lighting system. It employed Light Dependent Resistor LDR to sense the illumination level and compare the measured signal with the reference voltage realized through the usage of a potentiometer forming a potential divider. It highlights how an operational amplifier can be used as a comparator. This idea was transferred into Arduino Microcontroller. The latter is used to compare the signals using Arduino programming functions. The Arduinos multitasked using millis() to sound alarm at system switch-On or Reset. An I2C is used to interface a Liquid Crystal Display with an advantage of using two analog pins A4 and A5 instead of four or eight I/O pins for four and eight LCD modes respectively. The innovation will allow those I/O pins dedicated to other tasks plus its attendant simplicity. The usage of I2C scanner in identifying I2C programming address as 0x3F is explained. The step by step explanation of Arduino codes is an asset. The paper recommends among others further improvement in the display interactivity through robust programming. An alarm could also be made to sound at each switch-over. It is believed that its simplicity will encourage its mass production with its attendant socio economical benefits.

Keywords

Arduino; Microcontroller; Programming; Comparator; LCD; Lighting

Introduction

A control system can be defined as an interaction of components in order to form a system configuration which is required to produce a desired system response. While an Open loop control system has no feedback, a closed loop system has a feedback thus earning it the appellation of a feedback control system [1]. The latter is employed in automatic control of controlled variable. So it does not depend on an operator, a desired input is selected as reference, the output situation is sensed by a sensor, itself produces commensurate electrical signal which is compared with the signal generated by the reference input. A comparator compares the two signals; the resultant is the error signal that drives a controller which in turn controls the output. In order words, the error signal is fed to the controller to reduce the error and desired output is obtained [1]. In this case, the illumination of the environment is to be controlled, the sunlight illuminates the environment in the daytime while at night an automatic control system takes over to actuate a switch which switches on the light. The system is independent of an operator or the house owner. The issue of forgetfulness of an operator will not arise.

*Corresponding author: Oladimeji TT, Department of Electrical/Electronic Engineering, the Federal Polytechnic, Ado-Ekiti, Nigeria. E-mail: tolulopett@yahoo.com; Tel: +2348062919293, +2348030896393

Received May 02, 2018; Accepted May 09, 2018; Published May 22, 2018

Citation: Oladimeji TT (2018) Design and Implementation of Arduino Microcontroller Based Automatic Lighting Control with I2C LCD Display. SF J Telecommunic 2:2.

Copyright: © 2018 Oladimeji TT. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
Microcontroller

Nowadays, systems that have microcontrollers within them are common. They range from fans, TV remote controllers, Incubators to air-conditioning systems. There is widespread craves for these items by consumers due to their easy to control capabilities and compactness. Thus, hardly would you have electronic items in the house without microcontrollers.

A microcontroller is a small computer on a chip, unlike a microprocessor, it comprises processor, memory locations to store and execute programs. Apart from this, it consists of input and output ports to receive instructions and communicate with the outside world respectively [2]. A programmer programs microcontroller in high level language and a compiler compiles it and turn the instructions to low level machine language that can be understood by the controller. Computer understands ones and zeros, that is, two levels of HIGH (1) and LOW (0) which are binary digits. A compiled program written for a microcontroller on its IDE is expected to be downloaded into its ROM using a Programmer.

Methodology

The design project was consummated by highlighting the problems to be solved and an algorithm was spelt out to realize the solution. This was also buttressed using a flowchart thus simplifying the writing of the Arduino source codes. An engineer just need to install an Arduino IDE which is downloadable online on his computer from www.arduino.cc, write the program, compile it and load it into the board [4]. No need for a separate programmer that costs extra money. Its functions are easy to understand, a gesture which makes it ideal for prototyping by inventors. An Arduino uses C programming language [3].

Algorithms and Flowcharting

The following are the algorithms that list the sequence of steps needed to describe the solution to the design problem while the flowchart diagrammatically represents the order and interaction of activities and decisions.

Step 1: Read the values of $V_{\text{LDR}}$ and $V_{\text{REF}}$
Step 2: If the value of $V_{\text{LDR}} > V_{\text{REF}}$, output pin go HIGH (+5V) else LOW (0V)
Step 3: Buzzer State ==HIGH
Step 4: If $\text{millis()}$==$\text{Interval(mS)}$, Buzzer State should go LOW, else HIGH.
Step 5: Implement the Buzzer State
Step 6: If Outpin ==HIGH, Print “NIGHT: Power On” else Print “DAY: Power Off” (Figure 1)

Figure 1: The System Flowchart
Arduino Structures

The project uses the simplest Arduino board referred to as Arduino Uno [4] (Figure 2). It is a small microcontroller board. In this case an Atmega 328 that has 28 pins is used. Some of the components surrounding it are voltage regulator, reset button and Crystal oscillator; they allow the board to communicate with the computer without separate programmer. In order to enhance connections with the board, there are fourteen I/O pins; each pin is used as Input or Output pin depending on the specification in the codes written in the IDE. Also, there are six other analog input pins, A0 to A5, they take in Analogue quantity and convert them into a number between 0 and 1023. In the same vein, pins 3, 5, 6, 9, 10 and 11; though are digital can be programmed as analogue output pins depending on the codes written in the IDE.

An engineer can power the board from the computer USB port and can as well do that from a 9V battery wiring the tip of the plug as positive terminal. Regulated +5V terminals points and Grounds are added on the board for easy connections to other external circuitries.

Voltage Comparator

A voltage comparator compares two signals and produces an output based on which of the two is greater. Typical examples of comparator Operational Amplifiers (Op-Amps) are LM 311 and LM339. The output produced are either positive (HIGH) or Negative (LOW) saturation according to the difference of the input voltages [5] (Figure 3).

Two potential dividers are employed, one is formed from Light Dependent Resistor LDR and Resistor R1 as in figure 3 and other is formed by a variable resistor VR1 and R2. The variable is set in order to provide reference voltage depending on the level of darkness needed to switch on the light. The light sensor LDR senses the illumination (Controlled variable); its resistance at darkness is very high, about 10MΩ and about 100Ω at light.

Arduino UNO Based Voltage Comparator

An Arduino Uno Microcontroller armed with a well written code will perform the task indicated for a comparator. In this project Figure 4 Analog pins A0 and A1 are used. The output of the potential divider formed by the LDR and Resistor R1 is connected to A0 and the Output, slider terminal of potentiometer which represent the voltage reference is connected to A1. The two signals are compared using an if-else statements after being read by analog Read() function.

```c
val1=analogRead(0);
val2=analogRead(1);
//check which is greater
if(val1>val2){
digitalWrite(LED,HIGH);
}
else{
```

Figure 2: The Arduino Uno Card

![Arduino Uno Card](image)

Figure 3: LDR and Potentiometer Driving an Op Amplifier Connected as a Comparator

![LDR and Potentiometer Driving an Op Amplifier Connected as a Comparator](image)

Figure 4: An Arduino Uno Based Comparator
digitalWrite(LED,LOW); //Turn LED off
}

An analog Read (0) function reads A0, analog Read (1) reads A1 and assigned their values to Val1 and Val2 respectively. So if val 1 is greater than val 2, there is darkness; then digital Write the output terminal named LED HIGH with the statement: digitalWrite(LED, HIGH). If not, that is val2>val1 (Daylight), digital Write the output LOW using digitalWrite(LED,LOW). The low and high levels are 0V and +5V respectively. The high level will be used to actuate a transistor switch which has a relay operating coil as its load. The relay normally open contact will be employed to switch on the lighting units.

**Alarm using millis() Function**

This is a function that returns number of milliseconds Arduino has been up running, it overflows after about 50 days, that is, returns to zero (Smith, 2011). The syntax is millis(). It is used in this sketch to sound an alarm each time the device is reset or switched on. Already, an interval of 1000mS has been declared as an integer constant; while initial state of the buzzer has been fixed HIGH. So at switch ON there is HIGH Buzzer state.

```c
if(millis()>=interval){
    buzzer State=LOW;
}
else{
    buzzer State=HIGH;
}
digitalWrite(BUZZER, buzzer State);
```

When the mill is() is greater than or equal to the interval declared, the buzzer State becomes LOW, if not it becomes HIGH. The latter state of HIGH produces a HIGH value of +5V to the piezo electric device which sounds an alarm. The moment the condition of IF statement is met, buzzer State becomes LOW thus stopping the alarm. A digital Write statement: digitalWrite(BUZZER, buzzer State) writes Buzzer State(HIGH or LOW) to BUZZER pin. In this project, the buzzer is connected to an I/O pin 7

**I2C Liquid Crystal Display**

In this project, an i2C interface was used to interface Arduino Uno to the LCD. It allows only two analog pins A4 and A5 to be used. I/O Pins 2, 3, 4, 5, 6, 7 on Arduino can be used for other tasks. Another advantage is that one can connect more i2C devices on the same two analog pins to drive their LCDs. The i2C circuit board is compatible to various LCDs like 16x2, 16x4 or 20x4; the i2c backpack is connected to the back of the LCD. In this case, the microcontroller was connected thus:
1. VCC on i2C was connected to +5V on the Arduino
2. GND to GND
3. SDA to Data line A4
4. SCK on i2C to Clock A5

**I2C Scanner**

The code for initializing LCD with i2C interface was: `LiquidCrystal_I2C lcd(0x3F,2,1,0,4,5,6,7);` The address of the i2C device is 0x3F. i2C scanner scans the connection and returns 0x3F. The software was downloaded online from www.playground.arduino.cc. The code was copied and pasted on the IDE while the project is connected to the PC USB port COM 8. From the IDE Menu, Tools was selected and Serial Monitor opened. The window indicated 0x3F as the address of the i2C interface module thus: I2C device found at address 0x3F.

**Project Wiring**

The project was wired on a breadboard as shown in the Figure 5 below. The threshold of switching was set using the potentiometer in the potential divider which have its output feeding Arduino analog pin A1. As a flashlight is directed to the LDR, the LED switches off thus indicating day while LCD display displaying DAY: POWER OFF.
When the light from the flashlight is off, the LED comes on together with the LCD display showing NIGHT: POWER ON

**Codes**
```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
LiquidCrystal_I2C lcd(0x3F,2,1,0,4,5,6,7);
const int LED=13;
const int BUZZER=7;
int buzzerState =HIGH;
int val1=0; // to store state value of LDR
int val2=0; // to store the value of VREF
unsigned long interval =1000;

void setup() {
  lcd.begin(16,2); // for 16 X 2 LCD module
  lcd.setBacklightPin(3,POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.home();//go home
  lcd.print(" AUTOMATIC LIGHT");
  lcd.setCursor(2,1);
  lcd.print(" CONTROLLER");
  pinMode(LED,OUTPUT);
  pinMode(BUZZER,OUTPUT);
  //Note analog pins are automatically set as input
}

void loop(){
  val1=analogRead(0);
  val2=analogRead(1);
  //check which is greater
  if(val1>val2){
    digitalWrite(LED,HIGH);
  } else{
    digitalWrite(LED,LOW); //Turn LED off
  }
  if(millis()>=interval){
    buzzerState=LOW;
  } else{
    buzzerState=HIGH;
  }
  digitalWrite(BUZZER, buzzerState);
  delay(1000);
  if(digitalRead(LED)==HIGH){
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("NIGHT: POWER ON");
  } else{
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print(" DAY: POWER OFF");
  }
}(Figure 6)
```

**Conclusion and Recommendations**
The project has succeeded in showing simplicity in the use of Arduino in prototyping and designs of intelligent and embedded systems. The advantage of programming a microcontroller without a programmer coupled with the
Ease of debugging Arduino codes is an asset. In order to control lighting systems and other bigger loads with higher power expected to draw higher currents, Figure 6, a bipolar switching transistor or MOSFET driven relay can have its normally opened contact connected in series with the controlled lighting units [5]. In a situation where the current rating of the relay’s contact set is not enough, the latter can be connected in series with the operating coil of a contactor while the load, that is, the lighting load connected in series with its normally open poles.

MOSFET is being suggested being voltage operated, with very high input impedance can be driven directly by Arduino I/O pins. Ideas and functions used in this project can as well be employed in other equipment which has been hitherto designed with analog approach or other traditional microcontrollers. Here are other recommendations made:

i. Improvement can be made on display programs to allow interactive sessions with the user.

ii. Tone can be programmed to sound momentarily for a short period of time to alert the user on each transition, that is, switching on the lighting units and their switching off.

References
1. Saeed SH (2008) Automatic Control Systems. With Matlab Programs. 8th Edition. S.K. Kataria and Sons 1-3.

2. Bates PM. (2017) Programming 8-Bit PIC Microcontrollers in C Hardware Interactive Hardware Simulation, Newnes 1-3.

3. Arduino Tutorialspoint (2016) Retrieved online from www.tutorialspoint.com on 2017 24-35.

4. Banzi M (2011) Getting Started with Arduino, Second Edition. O’Reily Media 68.

5. Horowitz P, Hill W (1994) Art of Electronics. 2nd Edition. Cambridge University Press 230: 153-154.