Design of Automatic Fancy Lighting System

1. INTRODUCTION

1.1 OBJECTIVES AND SCOPE OF THE WORK

In this project work I am implementing an Automotive fancy lighting system based on LDR(light dependent resister)/ photo sensor and push buttons. Automatic fancy light control system needs no manual operation for switching ON and OFF when there is need of light it was ontrolled by photo sensor. It detects itself whether there is need for light or not. When darkness rises to a certain value then automatically light is switched ON and when there is other source of light i.e. day time, the light gets OFF. Along with photo sensor the system having two push buttons for manual operation, one for to control the briteness of RGD and second button for to on/off the light.

Nowadays every home/office already have a large number of electronic control systems. The complexity of the functions implemented in the automation systems necessitates an exchange of data between them. With conventional systems, data is exchanged by means of dedicated signal lines, but this is becoming increasingly difficult and expensive as control functions become ever more complex. In the case of complex control systems in particular, the number of connections cannot be increased much further. The relative simplicity of sensor interfaces make applications programming relatively simple in automation of home/office or any industry.

This proposed research work would be implemented using **Arduino** based embedded system design methodology, which includes Embedded hardware and firmware design modules . This project would be carried out with ATMEGA328P based Arduino UNO board and Aurduino driven Embedded EDA Tool kits.

Hardware requirements:
- ATmega328 Microcontroller based Arduino UNO Board
- Photo sensor
- RGB Led
- USB Cable
- Bread Board
- Connecting wires

Software requirements:
- Embedded C
- codebender Compiler
- fritzing.0.9.3b.64.pc

2. SYSTEM DESIGN

2.1 Block Diagram of UNO board

The block diagram of UNO board is as shown in Fig 2.1. It consists of power supply unit, microcontroller, Photo sensor, RGB-Led and two push buttons (switchs).

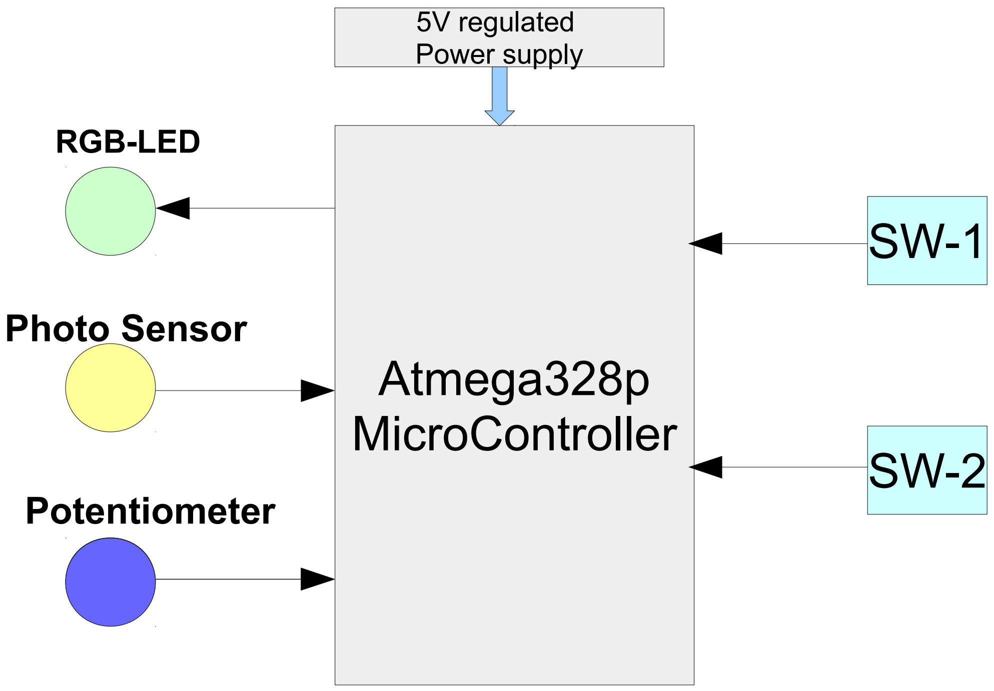
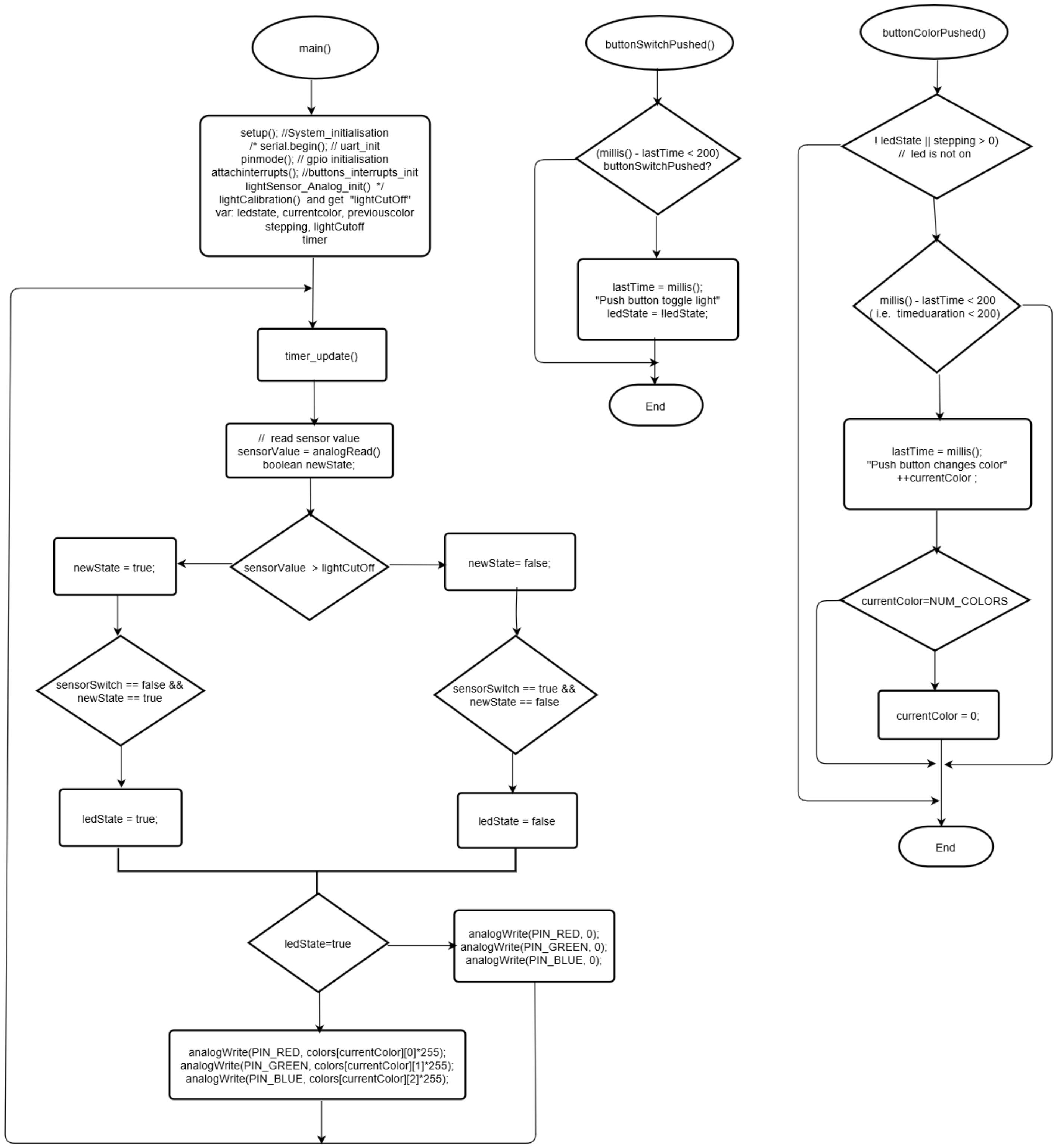


Fig.2.1 the Hardware diagram of the system

2.2 DESIGN OF SOFTWARE

In the project I use light detecting resistor as a light sensor, which is connected to analog input pin 'A0' of the Arduino board. The light sensor was precisely calibrated at system initialisation and calculates the lightCutOff (where lightCutOff = sensorLow + sensorHigh /2). Based on this lightCutOff we can decied the detection of high level or low level of voltage to energize the RGB Led. Along with light sensor, the system having two push buttons to control the RGB led manually. The Inturrupt service routines buttonSwitchPushed() and buttonColorPushed() was implemented to handle button pushes(buttonSwitch and buttonColor).



main()
setup(); //System_initialisation
/* serial.begin(); // uart_init
pinmode(); // gpio initialisation
attachinterrupts(); //buttons_interrupts_init
lightSensor_Analog_init() */
lightCalibration() and get "lightCutOff"
var: ledstate, currentcolor, previouscolor
stepping, lightCutoff
timer
timer_update()
// read sensor value
sensorValue = analogRead()
boolean newState;
sensorValue > lightCutOff
newState = true;
newState= false;
sensorSwitch == false && newState == true
sensorSwitch == true && newState == false
ledState = true;
ledState = false
ledState=true
analogWrite(PIN_RED, 0);
analogWrite(PIN_GREEN, 0);
analogWrite(PIN_BLUE, 0);
analogWrite(PIN_RED, colors[currentColor][0]*255);
analogWrite(PIN_GREEN, colors[currentColor][1]*255);
analogWrite(PIN_BLUE, colors[currentColor][2]*255);
buttonSwitchPushed()
(millis() - lastTime < 200) buttonSwitchPushed?
lastTime = millis();
"Push button toggle light"
ledState = !ledState;
End
buttonColorPushed()
! ledState || stepping > 0)
// led is not on
millis() - lastTime < 200
(i.e. timeduaration < 200)
lastTime = millis();
"Push button changes color"
++currentColor ;
currentColor=NUM_COLORS
currentColor = 0;
End

Fig.2.1: Flow Chart for Communication Unit

3. HARDWARE IMPLEMENTATION OF DASH BOARD

3.1 DASH BOARD

The implementation of the Dash board can be divided in two parts.

- Hardware implementation
- Firmware implementation

The **Hardware implementation** deals in drawing the schematic on the plane paper according to the application, testing the schematic design over the breadboard using the various IC's to find if the design meets the objective, carrying out the PCB layout of the schematic tested on breadboard, finally preparing the board and testing the designed hardware.

The block diagram discusses about the required components of Dash board and working condition is explained using circuit diagram and system wiring diagram.

3.1.1 Circuit Diagram of Dash board and Explanation

Fig 3.1 shows the circuit diagram of the Dash board. The circuit operates using 5V DC supply available through power adaptor/USB cable.

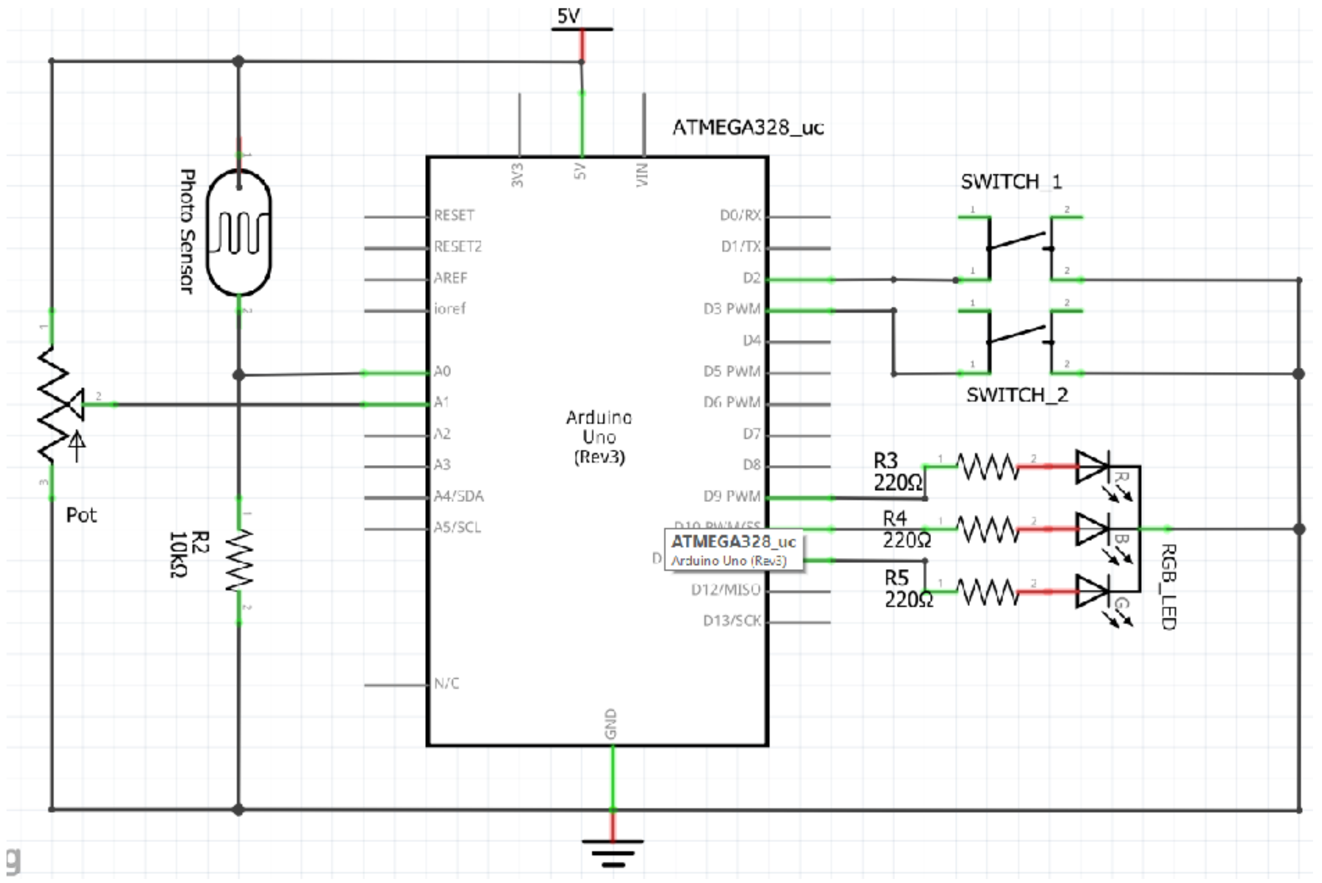


Fig 3.1: Circuit (Schematic) diagram of Dash board

The **Firmware part** deals in programming the microcontroller so that it can control the operation of the IC’s used in the implementation**.**

In the present work I have used the fritzing.0.9.3b.64.pc software for Schematic (circuit) design, the codebender Compiler to write and compile the source code, which has been written in the C language.

3.1.2 Connections of Modules:

PORT-D :

13	12	11	10	9	8	7	6	5	4	3	2	1
		R-Led	G-Led	B-Led						SW-2	SW-1	

PORT-A :

A5	A4	A3	A2	A1	A0
				POT	LDR

Fig. 3.2: Port Map for Monitoring Unit

1) RGB LED

Arduino pins 9, 10, 11 are connected to RGB LED +ve(anode) pins through 220E resister, and Arduino GND pin is connected to LED -ve(cathode) pin.

2) Push Buttons

Here in this project we are used to push buttons ButtonSwitch and ButtonColor. One End of ButtonSwitch was connected to Arduino 2 pin and ButtonColor was connected to Arduino 3 pin, other End of buttons are connected to Arduino GND pin.

3) Connections of Photo sensor:

Arduino +5v is connected to LDR One End and Arduino A0 pin is connected to LDR other end.
Arduino GND is connected to LDR other end with 10k resister.

4) Resistor

It is a passive component having two terminals that are used to manage the current flow in the circuit. A current that flows via a resistor is directly proportional to the voltage that appeared into the resistor. Value of resistance can be calculated with the help of multimeter or with the color code that is visible on the resistor.

Resistors are of two types :

i) Fixed Resistor – having a fixed value of resistance

ii) Variable Resistor – whose value of resistance can be changed for example if we have a resistor of 5K then the value of resistance will vary from 0 to 5 k.

3.2 INTRODUCTION TO HARDWARE MODULES

3.2.1 ATMEGA328 Microcontroller :

The ATmega328 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed. The ATmega328 contains 32K bytes On-chip In-System Reprogrammable fancy memory for program storage. The fancy memory has an endurance of at least 10,000 write/erase cycles.

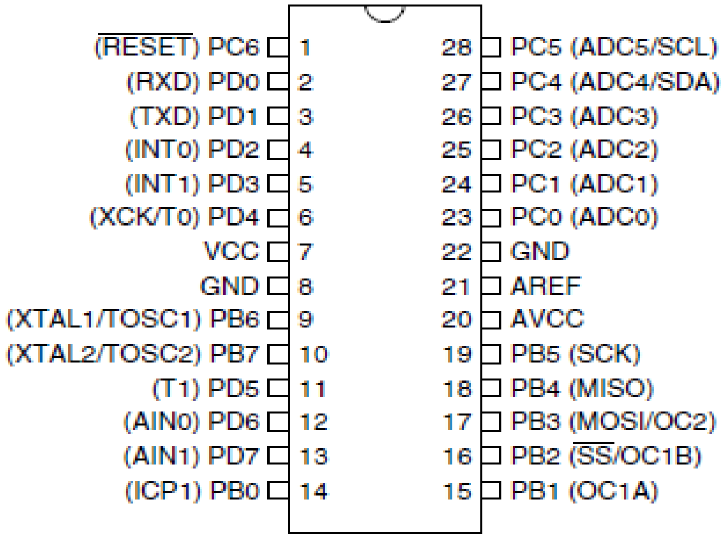


Fig 3.3: pin diagram of AVR Micro controller

3.2.2 Arduino's UNO Board :

An Arduino's **ATmega328** microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip fancy memory, compared with other devices that typically need an external programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, optiboot bootloader is the default bootloader installed on Arduino UNO.

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. The Arduino board is programmed via Universal serial bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR In-System programming (ISP) programming is used.

3.2.3 Analog-to-Digital Conversion (ADC):

The ATmega328 features a 10-bit successive approximation ADC. The ADC is connected to an 8- channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port C. The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. The ADC has a separate analog supply voltage pin, AVCC. AVCC must not differ more than ±0.3V from VCC. Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

ADC Conversion Result:

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH). For single ended conversion, the result is: ADC = VIN*1024/VREF Where, VIN is the voltage on the selected input pin and VREF the selected voltage reference.

3.2.4 USART Serial Communication

There are numerous serial communications protocols that can be used in data transmission. The ATMega328 has a full duplex serial port that allows data to be transmitted & received simultaneously in hardware while the software is doing other things. A serial port interrupt is generated by the hardware to get the attention of the program in order to read or write serial port data.

Both the transmit and receive buffers are accessed at the same location in the SFR space. The SBUF register. Writing to SBUF loads the transmit buffer, & reading from SBUF obtains the contents of the receive buffer. Register SCON controls data communication, register PCON controls data rates, & pins RXD (P3.0) & TXD (P3.1) connect to the serial data network.

3.3 SELECTION OF SENSORS

A sensor measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument. For example, a mercury thermometer converts the measured temperature into expansion and contraction of a liquid which can be read on a calibrated glass tube. A thermocouple converts temperature to an output voltage which can be read by a voltmeter. For accuracy, all sensors need to be calibrated against known standards.

Sensors come in many kinds and shapes to measure all kinds of physical variables. However, many sensors have some sort of voltage output. There are a number of implications to that. Here are some.

- If a sensor has a voltage output, then it is a voltage source that is controlled by the physical variable it measures. If the sensor is a voltage source, you need to remember that no physical voltage sources are ideal, and non-ideal voltage sources are usually best described with an Equivalent Circuit that contains the voltage source and an internal resistance.

- If a source has an internal resistance, there is a possibility of loading the source. If a significant load is attached to the source, the terminal voltage will drop. At that point, the terminal voltage is not what you expect it to be (from calibrations, spec sheets, etc.)

Need for Sensors:

- Sensors are omnipresent. They embedded in our bodies, automobiles, airplanes, cellular telephones, radios, chemical plants, industrial plants and countless other applications.
- Without the use of sensors, there would be no automation.

A good sensor obeys the following rules:

1. the sensor should be sensitive to the measured property
2. the sensor should be insensitive to any other property
3. the sensor should not influence the measured property

3.3.1 Light sensor

Also called photo resister or photocell or light Dependent Resistors(LDRs). LDRs are light-controlled variable resistors. The resistance of a photoresistor decreases with increasing incident light intensity. LDR is a device whose sensitivity depends upon the intensity of light falling on it. When the strength of the light falling on LDR increases the LDR resistance decreases, while if the strength of the light falls on LDR is decreased resistance increased. In the time of darkness or when there is no light, the resistance of LDR is in the range of mega ohms, while in the presence of light or in brightness in decrease by few hundred ohms. LDR have no polarity so you can connect any lid with leg. LDRs are very useful especially in light/dark sensor circuits.

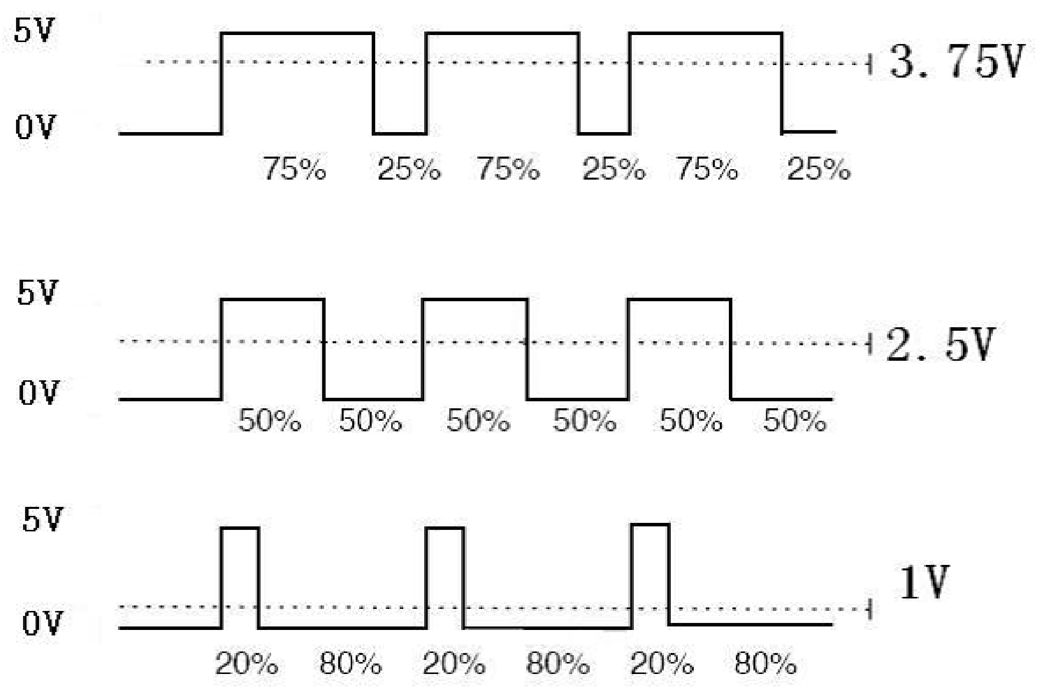
- When a light level of 1000 lux (bright light) is directed towards it, the resistance is 400 (ohms).
- When a light level of 10 lux (very low light level) is directed towards it, the resistance has risen dramatically to 10.43M (10430000 ohms).

3.5 RGB Led

The **RGB color model** is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue. Different devices detect or reproduce a given RGB value differently, since the color elements (such as phosphors or dyes) and their response to the individual R, G, and B levels vary from manufacturer to manufacturer. Thus an RGB value does not define the same color across devices without some kind of color management.

Using PWM Control RGB LED

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (3.3 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 3.3v controlling the brightness of the LED.



We can see from the oscillogram above that the amplitude of DC voltage output is 5V. However, the actual voltage output is only 3.75V through PWM, for the high level only takes up 75% of the total voltage within a period.

In PWM the term duty cycle describes the proportion of "on" time to the regular interval or "period" of time. Period describes the reciprocal of pulses in one second and the voltage amplitude is 0V-5V.

Here we input any value between 0 and 255 to the three pins of the RGB LED to make it display different colors.

5. SOFTWARE IMPLEMENTATION OF DASH BOARD

5.1 SOFTWARE TOOLS

For programming the microcontroller, the Arduino project provides an **integrated development environment** (**IDE**) based on the Processing project, which includes support for the **C** and C++ programming languages, I used embedded C code is used for project implementation. The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism to compile and load programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch". A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub main() into an executable cyclic executive program:

•setup(): a function that runs once at the start of a program and that can initialize settings.

•loop(): a function called repeatedly until the board powers off.

The system initialisation is done in setup() function :

- pin 2 and pin 3 as INPUT pins which are connected to push buttons (switches) named PIN_SWITCH and PIN_CPLOR and enabled the gpio interrupts for pin 2 and pin3.
- pin A0 and pin A1 as ANALOG input pins these are used to interface light sensor and potentiometer.
- pin 11, pin 10 and pin 9 are OUTPUT pins which are connected to the RGB led +ve(anode) pins 1, 3 and 4 through 220E resisters respectively.
- The lightCalibration() function used to calibrate light sensor at system initialisation, it calculates the lightCutOff (where lightCutOff = sensorLow + sensorHigh /2).
- The PotCalibration() function used to calibrate potentiometer at system initialisation, it calculates the PotCutOff (where PotCutOff = (PotLow + PotHigh) /2).

The working of circuit is implemented in the loop() function :

- If the button 1 is pressed it generates the inturrupt and goes to the interrupt attached function buttonSwitchPushed(), this function waits for switch debounnce and toggles ledState(0,1).
- If the button 2 is pressed it generates the inturrupt and goes to the interrupt attached function buttonColorPushed(), this function waits for switch debounnce and don't changes the led color if the light is not on or we are in the middle of stepping, else change the color gradually by linearly incrementing NUM_STEPPING.

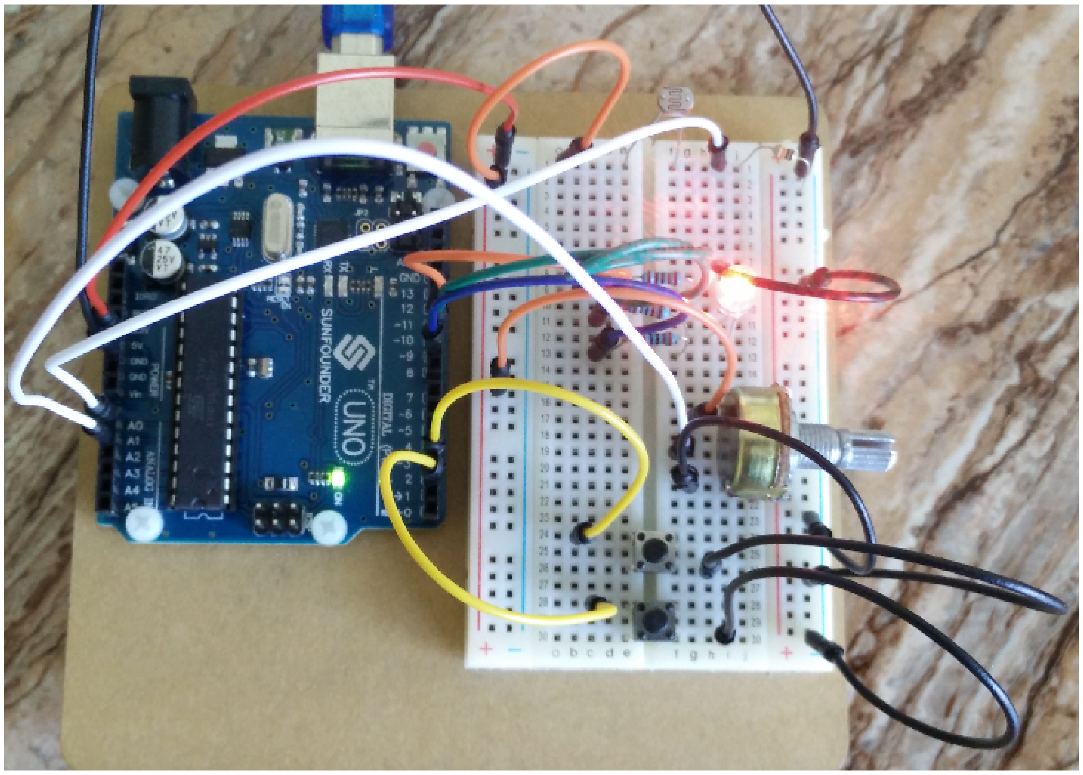
- CheckPhotoSensor() function reads the analog value on pin A0, which is connected to LDR, as the property of LDR that during the time of day resistance is low therefore voltage is high leades into the cutoff state which means LED will not glow. But in dimness or in night we know that resistance of LDR is high hence voltage low which make the LED to on state.
- checkPotentioMeter() function reads the analog value on pin A1, which is connected to pot. By rotating the pot we can change the resistance low to high as versa. If the resistance is low so we senses high voltage this leades into the cutoff state which sets ledState low. If the resistance is high hence voltage low which makes the ledState high.
- SetLedState() function sets the led on/ off based on ledState. If ledState is high, led is off, if ledState is low then generate the analog voltages on pins 11, 10 and 9.

The compilation of the sketch(C program) converts it into machine language file (.hex). This is the only language the microcontroller will understand. This hexadecimal code is loaded into the Arduino board by a loader program in the board's firmware.

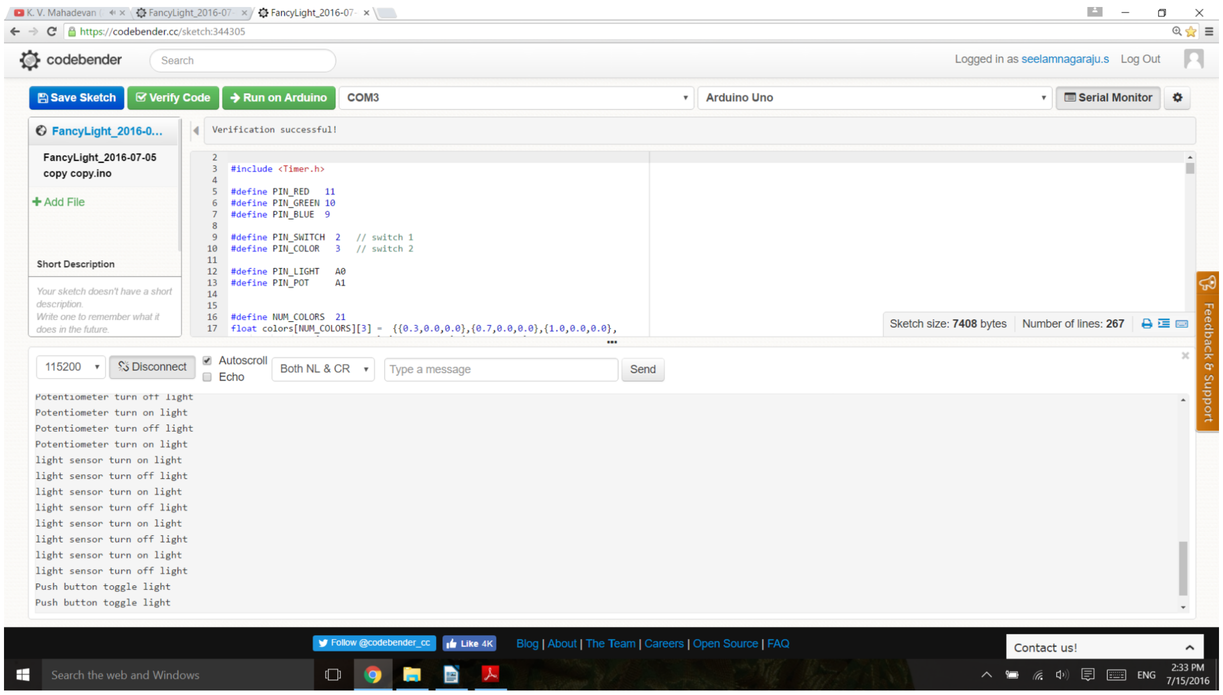
6. Appendix

- **Sketch URL :** https://codebender.cc/sketch:344305
- **Youtube video clip :** https://youtu.be/5rdZ1oNKJJ0

Hardware setup :



Serial console output of one complete run :



https://codebender.cc/sketch:344305

Logged in as seelamnagaraju.s

COM3 · Arduino Uno

FancyLight_2016-07-05 copy copy.ino

Short Description

Your sketch doesn't have a short description. Write one to remember what it does in the future.

```
Verification successful!
```

```

#include <Timer.h>

#define PIN_RED   11
#define PIN_GREEN 10
#define PIN_BLUE  9

#define PIN_SWITCH  2   // switch 1
#define PIN_COLOR   3   // switch 2

#define PIN_LIGHT   A0
#define PIN_POT     A1


#define NUM_COLORS  21
float colors[NUM_COLORS][3] =  {{0.3,0.0,0.0},{0.7,0.0,0.0},{1.0,0.0,0.0},
```

Sketch size: 7408 bytes Number of lines: 267

115200 · Both NL & CR · [✓ Autoscroll] · Echo

```
Potentiometer turn off light
Potentiometer turn on light
Potentiometer turn off light
Potentiometer turn on light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
Push button toggle light
Push button toggle light
```

One Complete session Run which demonstrate all the features of system :

```
-----------------------------------------------------------------------------------
Please vary light condition to start light calibration
Light cutoff value is 707
Please vary PotentioMeter to start POT calibration :
POT cutoff value is 511
Potentiometer turn on light
Potentiometer turn off light
Potentiometer turn on light
Potentiometer turn off light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
Push button toggle light
Push button toggle light
Push button toggle light
Push button toggle light
Push button toggle light
Push button toggle light
Push button toggle light
Push button changes color
Push button changes color
Push button changes color
Push button changes color
Push button changes color
Push button changes color
Potentiometer turn on light
Potentiometer turn off light
Potentiometer turn on light
Potentiometer turn off light
Potentiometer turn on light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
light sensor turn on light
light sensor turn off light
Push button toggle light
Push button toggle light
Push button toggle light
Push button changes color
```