Designing Embedded Systems with Arduino Microcontrollers: A Way Forward for Technological Advancement in Nigeria

Akinwole O.O.[1], Adeoye O.S.[2]

[1,2]*Department of Electrical and Electronic Engineering,*
The Federal Polytechnic, Ado-Ekiti, Nigeria

Abstract
Arduino Microcontroller has allowed many engineers to incorporate embedded systems in their designs due to its ease of use: There is no need to master many intricate computer technologies involving data lines, bits, instruction sets with their programming syntax and vocabularies. An engineer needs no other machine to program the Microcontroller apart from his desktop computer or laptop. Its open source status has allowed creation of various communities and groups on social media and elsewhere that do one project or the other using the Microcontroller. The paper provides short means of understanding the device by highlighting one-stop topics that new beginners would want to know. Arduino Uno Microcontroller structure is explained with its basic programming syntax and functions. As an example, a simple sequential timing system for an advertisement display was designed while its operational codes written and compiled on an Arduino Integrated Development Environment IDE while its simulation done on UnoArduSim V2.1. It is believed that this short communication will provide understanding for Electrical and Electronic engineers and enthusiast alike thereby providing enduring springboards for prototyping of Electrical and Electronic systems with its attendant socio-economic benefits.

Keywords: *Arduino Uno, Functions, Microcontroller, Programming, Sequential timing, Simulation*

.

I. INTRODUCTION

At the turn of twenty first century, there have been upsurge in number of equipment that use Microcontrollers; ranging from Television remote controls to fans and incubators that incorporates embedded systems to control amount of heat available for optimal hatching of fertilized eggs and its interactive display unit [1]. Microcontrollers offer better solution, being computers on single chips; enable production of embedded smart systems which are prevalent everywhere today [2]. A Microcontroller is a small computer on a single chip comprising a processor, memory and programmable input and output peripherals [3]. Thus, there are increase in craves for the new knowledge by engineers and students alike, especially in the areas of prototyping power electronic devices in order to save time, improve efficiency and enhance artificial intelligence. In the same vein, Microcontroller topologies are being developed and improved upon to meet the yearnings of design engineers. There are PIC microcontrollers, the engineer need to understand their architectures in relation to their data lines, memory locations and their instruction sets. The knowledge of digital electronics in relation to conversion from one base to the other will enable him to programme them. Arduino Microcontroller is an open source prototype platform that simplifies the art of designing embedded systems; it comprises a board referred to as an Arduino board and can be programmed in its Integrated development environment IDE and the compiled sketches uploaded into the board through a Computer USB without the agency of a programmer [4]. Since not all computers have serial ports, so it is an advantage. Arduino platform, being an open source, has numerous groups working on it , in other words an engineer can source circuits diagrams online with their codes and assemble with little or no modification without fear of repercussions from copying. The system was born to enhance interactive designs which allow prototyping a centre of its activities [5].

II. ARDUINO UNO MICROCONTROLLER STRUCTURE

An Arduino Uno Microcontroller comprises hardware and software as its major parts. It consists of a PIC Microcontroller ATmega328P as its main component [6]; the mapping of the integrated circuit pins in relation to Arduino pins is shown in figure 1.0. The software is realized on its Integrated Development Environment IDE, the sketch is the small program or codes that are downloaded into the board after a successful compilation which directs the microcontroller what to do. The hardware has fourteen digital input-output pins (0-13), the sketch specifies whether a pin is to be recognized as an input or an output. Also it includes six analog inputs pins, (A0-A5), with them, ever changing analog values can be read from sensors like a Light Dependent Resistors LDR, Thermistors and Potentiometers, the values

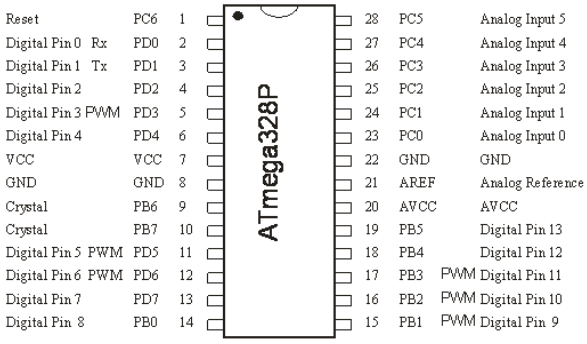


Figure 1.0: Arduino Pins Mapping

obtained are converted to a number between 0 and 1023 (2^{10}-1). Arduino Pins 3,5,6,9,10 and 11 though are digital I/O pins can be used as Analog output pins, it all depends on codes or sketch a design engineer creates on the IDE. For the power requirement, the board can be powered from a 9V battery while wiring the centre terminal of barrel type plug as positive (+Ve). In the course of prototyping, power can also be sourced from the computer's USB port.

III. ARDUINO PROGRAMMING

An Arduino program comprises two main parts which are the Setup () and the loop () functions. The former is normally employed to initialize the variables to be used and setup the board. It is expected to run once each time power is switched on or at each reset action. The latter which is the loop () function allows consecutive loops; it controls the board actively. The word void means no information is expected to be returned for functions from which it was called [4]. Thus void loop () contains the main codes of an engineer's sketch, it includes instructions to be repeated continuously until the board is disengaged from the power source. Every line of the sketch is normally terminated by a semicolon.

IV. SOME ARDUINO FUNCTIONS

Arduino functions are referred to as series of statements during programming actions which programmers can called by their names. Below are examples of basic Arduino functions.

1. *pinMode() Function*

This function is used in setting up a specific pin whether it should work as an input or as an output pin: The syntax is

```
void setup()
{
pinMode(pin,mode);
}
```

Where pin refers to the number of the pin upon which to set as input or output while mode refers to the input or output for example **pinMode(9, OUTPUT);** means Arduino pin 9 is to be an output pin.

2. *digitalWrite() Function*

digitalWrite() function is normally employed to write a HIGH logic to a digital pin. If a pin is digital written, that pin becomes +5V for HIGH and 0V for LOW

3. *digitalRead() Function*

A digitaRead() function is used to read voltage levels on Arduino pin. There are two levels, HIGH (+5V) and LOW (0V). The state of the pin be used to perform a logical operation using IF statement :

```
void loop()
{
if(digitalRead(9)==HIGH)
digitalWrite(5,LOW);
}
```

If the value of pin 9 HIGH (+5V), then pin 5 should be lowered to LOW (0V).

4. *analogRead() Function*

The syntax is analogRead(pin); Arduino has six analog pins, that is, A0-A5, the pins can be read using the function. Arduino pins Works with the maximum of 5V, so a 0V to +5V, the microcontroller returns a value from 0 to 2^{10}-1 (1023). In actual fact, a 2.5 V on an Analog pin A5 corresponds to 512. In the figure 2.0 below a potential divider is formed by resistor R1 and LDR, its output V_{LDR} is connected to pin A0 while a reference voltage Vref derived from the potentiometer is connected to pin A1. As darkness comes, V_{LDR} rises, the situation can be used to take the output pin to HIGH (+5V) which can be used to drive a transistor switch for the switching ON of Lighting units [1].

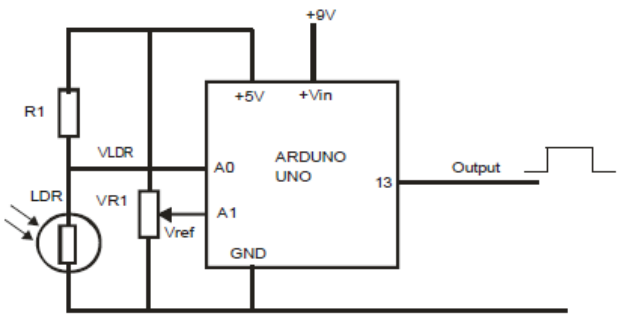


Figure 2.0: Analog Pins A0 and A1

```
void loop()
valA=analogRead(A0);// Read pin A0 and store the value at memory location valA
valB=analogRead(A1);//Read pin A1 and store the value at memory location valB

if(valA>valB) // To check which pin has greater voltage level
  {
```

```
digitalWrite(13, HIGH);
}
else
{
digitalWrite(13,LOW);
}
```

5. delay() function

A delay function causes the implementation of Arduino codes to wait for a period of time before moving to the next line. The syntax is **delay(mS).** The time is specified in millisecond. For example, delay(1000) means the system should wait for one second. Sometimes, for convenience sake, delayMicroseconds() function can be used, in this case, the time is in Microseconds. A delay of 1.6mS cannot be written as delay(1..6) but delayMicrosecond(1600). For timing purpose, delay function is not ideal because it is a blocking function [4] it causes all other activities within the icrocontroller to stop instead a millis function is usually considered. The program below blinks an LED connected to pin 13. Apart from the external LED that can be connected to pin 13, Arduino board has an LED on the it connected to pin 13. So an engineer can use the LED especially during programming.

```
int  LED =13;
void setup()
pinMode(LED, OUTPUT);// LED is set to be
                        output pin
void loop()
{
digitalWrite(LED, HIGH);// LED is lit
delay(2000);// wait for 2 seconds
digitalWrite(LED,LOW);//LED is switched off
delay(2000); //wait for 2 seconds
}
```

6. millis() function

A millis function do returns number of millisecond the system has been up running, if the microcontroller is switched on for 50days, it will overflow, that is , it will return to zero; in other words it overflows after 50days of running. For example,

```
const int  LED =9;// pin 9 is called LED, an LED
                    anode is connected to it.
const int interval= 2000;//an interval of time is 2
                          second
int LED=HIGH; //LED  Anode is set High (+5V)

        void setup()
        pinMode(LED, OUTPUT);// pin 9 having
                LED connected to is set to be an
                output.
        void loop()
        {
        if(millis()>=interval)// if the time counted by
millis function is greater or equal to interval(2s)
        {
       LED=LOW;// LED that is hitherto HIGH(lit)
becomes LOW(off)
        }
        else
        {
        LED=HIGH;// Remain lit
        }
        }
```

7. Liquid Crystal Display

Arduino can be connected to liquid crystal display using 8 bit or 4bit mode, the figure 3.0 below shows a four bit mode where RS (Register select) pin on the 16 X 2 LCD is connected to digital pin 4, it controls the location of the LCD's memory where Arduino will write the current character. E (enable) pin of the LCD is connected to pin 5, that is, it allows writing into the aforementioned registers. R/W (Read/Write) pin is connected to the ground or to the Arduino

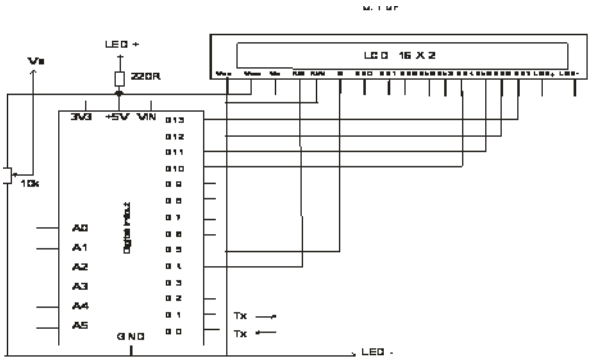


Figure 3.0: Arduino Board-LCD Interfacing

digital pin 10 which is optional. In case you are planning to get a feedback from the LCD, connect to pin 10 if not to GND. Thus in 4 bit mode, DB0-DB3 need not be connected [7].
The syntax is LiquidCrystal lcd(RS,EN,D4,D5,D6,D7), in this case: LiquidCrystal lcd(4,5,10,11,12,13) are connected to RS,EN,D4,D5,D6 and D7 respectively.

//Simple Arduino Code to write to the LCD

```
 # include<LiquidCrystal.h>
LiquidCrystal lcd(4,5,10,11,12,13);
        void setup()
        {
        lcd.begin(16,2);//for 6 x 2 LCD module
        lcd.home();//go home
```

```
lcd.print("Designing System");// Print
                                message
}
Void loop()
{
lcd.clear();//Clear the screen
lcd.setcursor(1,0);//set the cursor at column
1                   and row 0
lcd. Prnt("AKIN OYE FPA");
delay(500);
}
```

8. I2C LCD Interface

With I2C interface, only two analog pins A4 and A5 are employed. The digital pins can be available for other tasks. It has another advantage of allowing an engineer to connect more devices on the same two pins to drive their LCDs. It is compatible to various LCDs like 6 X 2 and 20 X 4.The connection is made as follows:

- +5V on I2C connected to +5V on the Arduino
- Ground (GND) to GND of the Arduino board
- SDA on I2C board to A4
- SCK on I2C to A5

V. A DESIGN EXAMPLE

Four light emitting diodes are connected to pins 2, 3, 4 and 5 through current limiting 220Ω resistors. A fixed voltage regulator 7809 is used to generate +9V DC [8] supply and fed into Vin terminals. (Figure 4.0). After compilation of the codes in the Arduino IDE, these tokenized codes are transferred into the board right from the computer system without an agency of separate programmer. Table 1 shows the states in relation to the lighting of the LEDs.

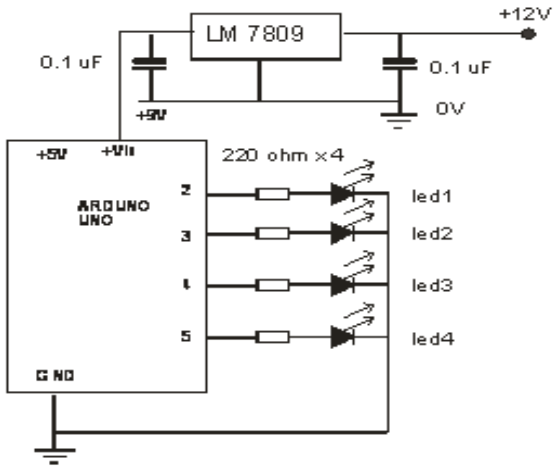


Figure 4.0: The Project Circuit

TABLE 1

Sequence of LEDs Operations

		STATES							
	COLOUR	1	2	3	4	5	6	7	8
LED on Pin 2	RED	ON	ON	ON	ON	ON	ON	ON	OFF
LED on Pin 3	YELLOW	OFF	ON	ON	ON	ON	ON	OFF	OFF
LED on Pin 4	BLUE	OFF	OFF	ON	ON	ON	OFF	OFF	OFF
LED on Pin 5	GREEN	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF

VI. ARDUINO CODES

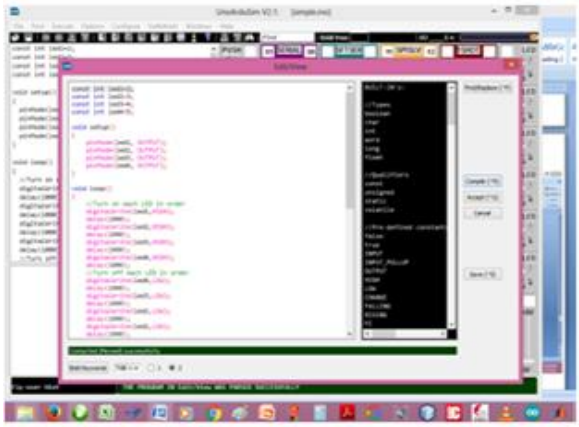
```
// Program to generate light pattern

const int led1=2;// pin 2 is named led1
const int led2=3; // pin 3 is named led2
 const int led3=4;// pin 4 is named led3
const int led4=5;// pin 5 is named led4

void setup()
{
pinMode(led1, OUTPUT);// led1 should be
recognised as an Output pin
pinMode(led2, OUTPUT); // led2 should be
recognised as an Output pin
pinMode(led3, OUTPUT); // led3 should be
recognised as an Output pin
pinMode(led4, OUTPUT); // led4 should be
recognised as an Output pin
}
void loop()
{
//Turn on each LED in order
digitalWrite(led1,HIGH);// turn ON LED on
                        led1
delay(1000); //wait for 1s
digitalWrite(led2,HIGH); // turn ON LED
on                      led2
delay(1000); //wait for 1s
digitalWrite(led3,HIGH); // turn ON LED
on                      led3
delay(1000); //wait for 1s
digitalWrite(led4,HIGH); // turn ON LED
on                      led4
delay(1000); //wait for 1s
//Turn off each LED in order
digitalWrite(led4,LOW); // turn OFF LED
on                      led4
delay(1000); //wait for 1s
digitalWrite(led3,LOW); // turn OFF LED
on                      led3
delay(1000); //wait for 1s
digitalWrite(led2,LOW); // turn OFF LED
on                      led2
delay(1000); //wait for 1s
digitalWrite(led1,LOW); // turn OFF LED
on                      led1
delay(1000); //wait for 1s
}
```

11. Design Simulation

Arduino platform, being an open source type has attracted more engineers, technologists and technicians into design of embedded systems, today, virtually all areas of technologies have witnessed unprecedented growth in knowledge. This has caused many others in creating simulators for the Arduino board. Proteus can be used for simulation of designs that employ the microcontroller. In this work, the simulator named UnoArduSim V2.1 was used [9] (figure 5.0a) In its environment, the menu was selected, edit view clicked, codes written above typed in and compiled. A successful compilation message was returned thus "Compiled (Parsed) Successfully". LEDs on the simulator was connected to pin 2, pin 3, pin 4 and pin5 while LEDs colours were selected as R,Y,B and G; (Red, Yellow, Blue and Green). RUN or F9 key was used to execute. See the result in figure 5.0b



(a)

Figure 5.0 (b): Arduino Board-LCD Interfacing

VII. CONCLUSIONS

The paper has succeeded in shedding light on Arduino microcontroller platform in relation to its structure, programming functions, wiring and simulation. The design example controls four LEDs sequentially. The project can be employed in implementing advertisement display system where each segment can be lit by a drive signal which hitherto controls low power LED. Since a maximum of 20mA [1] could be drawn from the board, LED pins led1, led2,led3 and led4 can be arranged to drive electronic switch having relay as load, [10] the normally contact set will now control lighting units that draws current even in excess of 10A. The design, when modified can be used to generate signals controlling an active HIGH industrial time based sequential system like Cataphoresis plant robotic system. In such system, An LCD function would be included to display the current status of its operations. Those HIGH periods can be varied the duration of current machine process. The following recommendations are made:

1. More awareness should be created in the country on the advantages of the platform thereby accentuating creation of prototypes.
2. Schools' curriculum should be updated to include Arduino platforms.
3. Capacity building workshops should be organized on Arduino Microcontrollers
4. Arduino Microcontroller based embedded system should be encouraged in student's projects.
5. Modification of existing underperforming system can be achieved using the platform.

REFERENCES

[1] Akinwole, O.O and Oladimeji, T.T "*Design and Implementation of Arduino Microcontroller Based Automatic Lighting Control with I2C LCD Display*". J. Electr ElectronSyst 7:258, 2018

[2] Bates, M.P.,"*Programming 8-bit PIC Microcontrollers in C with Interactive Hardware Simulation*," NewnessUSA, 2008.

[3] Smith, A.G. "*Introduction to Arduino: A Piece of Cake*", 2011 Retrieved online 2018 from www.intotoarduino.com

[4] Tutorialpoint. 2016.Retrieved online from www.tutorialpont.com, 2018, 24-35

[5] Banzi, M. **"***Getting Started with Arduino*" , 2nd Edition O'Reily Media, 2011

[6] Schwartz, M., "*Arduino Home Automation "Packt Publishing*, UK, 2014

[7] HandsonTechnology**,"** *Learn Arduino-LCD Interfacing*". 2018 Retrieved online 2018 from www.handsontec.com

[8] Horowitz, P., Hill, W., "*The Art of Electroni.cs*", Cambridge University Press, 1989

[9] Simmon, S **"***Arduino Simulator*". UnoArduSim V2.1. 2018 Downloaded online 2018 from www.sites.google.com

[10] Akinwole, O.O." *Design and Simulation of an Arduino Based 1kVA Modified Sine Wave Inverter using Proteus*". J. Electr Electron Syst 7:258, 2019.

[11] J Vignesh, V Nijanthan, J Venkateshwaran, K Suresh Kumar and Mrs B. Vidhya" *Digital Fuel Level Indicator for Motor Bikes using Arduino Microcontroller*". SSRG International Journal of Electronics and Communication Engineering – Volume 4 Issue 3 –2017.

[12] Nicodemus M. Sakayo1 , Joseph N. Mutuku2 , James M. Ngaruiya3" *Design and Calibration of a Microcontroller Based MQ-4 Gas Sensor for Domestic Cooking Gas System*" .SSRG International Journal of Applied Physics– Volume 6 Issue 2– May to August 2019.