AN ARDUINO BASED BUILDING LIGHTS MANAGEMENT SYSTEM

Sajid M. Sheikh¹, Tshoganyetso Mangadi²

¹,²Department of Electrical Engineering, University of Botswana, Gaborone, Botswana

ABSTRACT

Lights in Building use a significant amount of power of the overall building energy consumption. With the power loading problem faced in Botswana and many other developing countries, Building Light Management systems can help save power. The Building Lights Management System presented in this paper is based on the Arduino Board. The choice of the Arduino board from other Open source microcontrollers was done based mainly on the programming environment, libraries, choice of a ready to use microcontroller, low cost and future system enhancements. This paper presents a general design of a Low cost Building Lights Management System (BLMS) using an Arduino microcontroller. The hardware design, software designs and the arduino c++ code are presented in this paper. This basis of this design can be used to develop any BLMS arduino based system.

Keywords: Microcontrollers, Arduino, Raspberry PI, Hardware Design, Software Design, Lights Management.

I. INTRODUCTION

The Lighting Market Characterization study is a multiyear program to evaluate light sources in the United States, and identify opportunities for saving energy. Phase I of this study, provided an estimated inventory of installed lighting technologies for 2001 and their associated energy consumption. It was shown that lighting used approximately 8.3% of national primary energy consumption, or about 22% of the total electricity generated in the U.S. Thus electricity used for lightening is a significant amount [1].

Botswana currently is facing power demand problems and has to use the solution of Load shedding at times to address the problem. A similar problem is faced in many developing countries. If lights are used effectively and are on only when needed, this can help solve some of the supply shortage issue. Botswana has a national electricity demand of 500MW. To fulfill this need, Botswana operates the Morupule Power Station, which contributes only 20% (120MW) of the total.
balance of 80% is imported mainly from ESKOM in South Africa, Hydroelec terica de Cahora Bassa (HCB) and Electricidade de Mocambique (EDM) of Mozambique. The Southern African region as a whole is experiencing power supply deficit and most of the power utilities in the region, including BPC are grappling with the supply deficit. As a result, neighbouring countries are rapidly reducing exports to Botswana, which was forced to resort to load shedding since 2008. Botswana is currently pursuing the Morupule B project to achieve energy generation self-sufficiency. It is estimated that the Morupule B project will increase household access rates from 47% to 80% by 2016, which is in line with Botswana’s energy strategy of increasing energy access to all its citizens [2].

In a research done on the “Analysis of Electricity Consumption for Lighting and Small Power in Office Buildings” taking a case study approach, the paper focused on the energy performance of two multi-tenanted office buildings located in London and Bristol. It was concluded that lighting accounts for 24% of the annual electricity consumption as can be seen in figure 1 [3].

![Figure 1: Electricity Consumption Breakdown of a Building][1]

Lighting controls not only offer electricity savings, but also offer further benefits depending on the application they can improve comfort, reduce maintenance costs and impart greater flexibility to the use of a workspace [4].

Lighting controls are best deployed as a reliable means of turning off the lighting. People will turn lights on when they need them; sometimes they forget to turn them off. They are some effective devices that control the automatic switching on and off of lights. These include movement sensors such as Passive infra-red (PIR), Ultrasonic or Microwave, daylight sensors and timers [4] [5].

In a lot of corporate buildings in Botswana you find that during the day a lot of lights that should be turned off are still on. Examples of these lights are external lights, bay lights, car park lights, streetlights etc. In a lot of cases, these lights are meant to be physically turned on at night and off in the morning by a person earmarked for this job, and in some other cases there is supposed some sort of automatic ON/OFF device to control the turning of these lights at night and during the day. However, both of the arrangements do not work well as evidenced by a lot of lights that remain on during the day. In the case of manual systems most people simply forget to put off these lights in the morning, or are simply not adequately trained to do so. In the case of automated systems, the devices employed malfunction easily and are also unintelligent in that they are simple electro-mechanical devices which cannot easily report failures and also do not incorporate other features, besides day-light sensing, to detect when to turn on/off the lights.

External building lights generally consume a lot of energy and a lot of energy (and money) is wasted by corporations when these lights remain on during the day. Besides the money that is unnecessarily lost, a lot of energy is wasted in an era where energy saving is very important and emphasized at both local and national level. Saving energy allows cheaper supply of energy, burning less coal and saving the environment via less emissions and lesser use of water.
This paper presents a general design of a Building Lights Management System (BLMS) using an Arduino microcontroller. The hardware design, software designs and the arduino c++ code are presented in this paper. This basis of this design can be used to develop any BLMS Arduino based system.

II. OBJECTIVES

The proposed Building Lights Management System will control the operation of external building lights by ensuring that lights are automatically turned ON at night and turned OFF in the morning. The system will be an intelligent system incorporating other features not commonly found in current systems. These are:

- Automatic day-light sensing to turn on/off light.
- Sensing whether light is actually ON/OFF.
- Time-of-day tracking to further control lights operation.
- Controlling which lights turn on. Not all lights may need to be operated at all times e.g. summer vs winter
- Central panel where operation of all lights is controlled and status is viewed.

III. MICROCONTROLLERS

A microcontroller is a small computer on a single integrated circuit and are specifically designed for embedded applications. They are many Open source microcontrollers available in the market such as Arduino, Resberry PI, Rascal, ARM mbed, pcDuino, Cubieboard, Beagle Board, MintDuino, Netduino and Beagle Bone[6]. A brief comparison with a few of them is given in Table 1 below.

| Table 1: Comparison of some Open-source Microcontrollers [6] |
|----------------|----------------|----------------|----------------|----------------|
| Board:         | Arduino Uno   | MintDuino     | Netduino Plus | Raspberry Pi   |
| Approximate   | $30           | $25           | $60            | $40            |
| Price         |               |               |                |                |
| Special       | Onboard USB   | DIY Arduino!  | Programmed with .NET Micro Framework; Onboard Ethernet | HD Capable Video Processor, HDMI and Composite Outputs, Onboard Ethernet |
| Features      | controller    |               |                |                |
| Processor     | ATmega328     | ATmega328     | STMicro 32-bit microcontroller | ARM1176JZF-S |
| Speed         | 16 MHz        | 16 MHz        | 168 MHz        | 700 MHz        |
| Analog Pins   | 6 (6 PWM)     | 6             | 22 (GPIO - digital or analog) | 8 (GPIO - Digital and Analog) |
| Digital Pins  | 14 (6 PWM)    | 14 (6 PWM)    | 22 (GPIO - digital or analog) | 8 (GPIO - Digital and Analog) |
| Memory        | SRAM 2KB - EEPROM 1KB | SRAM 2KB - EEPROM 1KB | 384KB Code Storage | RAM 512MB |
| Programmed    | Arduino / C Variant | Arduino / C Variant | Microsoft .NET Environment | Any language supported by a compatible Linux distribution (such as Raspbian or Occidentalis)** |
| Language      |                |               |                |                |
|                |                |               |                |                |

**Includes Angstrom Linux on SD Card. Any language supported by a compatible Linux distribution (such as Ångström or Ubuntu)
a. Raspberry PI: The Raspberry Pi is a low-cost credit-card sized computer. It was developed for use in teaching computer programming to children, who are now using it in schools, at code clubs and at home. Its users have given it thousands of other applications, from near-space weather balloons to baby monitors to industrial applications, and its cheapness, robustness and low power consumption have sparked interest in the developing world. [7]

The Raspberry PI could have been used in the design of this BLMS system, but it is more specialized towards user interface applications.

b. Arduino: The Arduino is one of the latest embedded system quick prototyping and development platform, which is very popular with a large community of embedded system developers, students and hobbyists. The huge popularity of the Arduino boards concept over other traditional prototyping and development boards is that the Arduino also comes with a much more simplified C-based software development platform which makes easy and quick prototyping very simple. It also similarly allows people who are not highly trained in C programming language to learn and develop easily. These people include a large community of hobbyists and students who have different background to that of electronics and software engineering [8].

Another huge advantage of the Arduino board is that the software development tools and software libraries are free and open source. The electronic designs of the board (schematic and pcb) are also open source. This allows a developer to work very cheaply and quickly when experimenting with new technologies and concepts. For instance, a lot of libraries for easy manipulation of digital i/o, analog pins, serial communications, math come freely out of the box with the Arduino IDE. Other libraries covering more complex operations are active developed by the user and open-source community, these include GPS communications, motor drives, SPI, Ethernet, wifi etc. Furthermore the Arduino system also has a lot of shields, which are hardware modules that one can use to augment the basic board to perform more complex functions. Most of the shields are commercial but their designs are open-sourced meaning that you can build them for yourself and customize them [8][9][10].

The main advantage of the Arduino over the Raspberry PI and a few other microcontrollers is the easy to use programming environment. The Arduino uses a simplified version of C++ with libraries developed for many different applications and equipment. With the Arduino board, the digital and analog pins are ready to use with headers already broken out. Other microcontrollers such as the Raspberry PI requires a specific Circuit Board (PCB) to be designed and placed onto of the microcontroller.

These BLMS project has chosen to use the arduino system due to the above-mentioned advantages. This will allow the system to be developed very fast, cheaply and with a lot of flexible. For instance, in the second version of the product, it might be necessary to use an Ethernet based system in order to have multiple controllers distributed across the building. Then using an Arduino it will be a simple matter of using the Ethernet shield and the open source Arduino TCP/IP stack library to ease development.

IV. METHODOLOGY

The following general methodology is proposed for building the Arduino based Building Lights Management System (BLMS) as shown in figure 2.
a. System Requirements and Specification

Since the BLMS is not a trivial system, it is important that requirements are gathered beforehand in order to inform the design evolution of the system. This step also specifies the context and important parameters of the system, including number and power rating of lights to be controlled, user interaction features, communication modes etc.

b. System Design – Hardware Block

The general organization of the hardware system is developed in this step. The system context captured in step (a) is also further refined and proposed interconnection of various modules is depicted by way of a graphical block diagram.

c. System Design – Software Block

Following step (b), a similar overall software organization to run in the hardware above is captured. It is to be noted that this is an overall organization of how the software will be implemented in the context of the hardware above and therefore follows closely the way the hardware is organized. It is therefore the overall software architecture of the system.

d. Detail Design – Hardware Schematics and PCB

Having agreed on the hardware block organization, it can be proceed to capture the schematic drawings and the subsequent printed circuit board (PCB) which comprises the finalized hardware design for the system. Please note that in the case of Arduino development this may mean also developing custom shields if an appropriate shield cannot be obtained off-shelf.

e. Detail Design – Software

Following the development of the overall software block, it is necessary capture the organization of the modules, functions and important variables in more detail, and this is done in this step.

e. Implementation – Coding

In this step, the actual software coding is done and when using the Arduino, a lot of special attention is paid to using the extensive Arduino libraries which make it easier and fast to write programs.
f. Testing and Deployment

Any electronic and software system under development has to be thoroughly tested before it is deployed. It is common to have a highly iterative cycle of step (e) – (f) and with the Arduino board this iteration can be very fast due to the rapid prototyping allowed by the arduino platform. It can be important to deploy the system even as part of testing to ensure that the interaction between hardware and software is as per the design as well as to test the system against the system requirements and specification.

It is to be noted that not all steps are depicted in this paper, since the objective is to show conceptual steps geared towards implementation. It was necessary to omit steps (a), (d) and (f). The other steps are highlighted below pertaining to the implementation of the ABLMS, with the section II (OBJECTIVES) used as the system requirements.

V. RESULTS

1. SYSTEM DESIGN (HARDWARE BLOCK)

The proposed solution is to use the arduino uno as the heart of the intelligent control system and develop the associated hardware and software to implement the system. The proposed system architecture is captured in figure 3.

![Hardware setup for the Arduino based BLMS system](image)

**LIGHT:** The lights are the units to be controlled. Up to 10 individual lights can be connected however the demo unit may use only one to two lights. The controller operates the relay / contactor that turns the light on/off. In a system where it is critical, an additional low light sensor may be used to ensure that the light is on. The lights are individually controllable.

**DAYLIGHT SENSOR:** It is used by the controller to know when it is day-time and when it is night time. A built-in time of day clock also complements this sensor and both are used to know when to turn the lights on or off. Additionally, it may be necessary to turn lights on/off out of sync with the daylight sensor e.g when they need to be turned for only 5 hours or so to save power, therefore the timer may be used for this.

**MODESET BUTTON:** The user presses this button to when they want to change the mode from daylight sensor to time-based mode and vice versa.
SYSTEMSET BUTTON: The user presses this function when they want to setup the system.

REAL-TIME CLOCK: It is used by the controller to know when it is day-time and when it is night time. A built-in time of day clock also complements this sensor and both are used to know when to turn the lights on or off.

USER: Programs the system and observes the status of the system, for example

- Set which lights are to be controlled, if not all.
- Ensure the built-in clock is set correctly.
- Observe status of lights (whether on/off) via local LEDs on control panel.
- Program timing of individual lights, if not sync’ed with daylight.

From the Preliminary Hardware design, the software designs are developed, the C++ code written in the Arduino compiler, code compiled and then loaded to the Arduino board. The system is then tested with the system programmed. The Arduino board allows more modules to be added to the system at a later stage and also the system to be reprogrammed.

2. SYSTEM DESIGN (SOFTWARE BLOCK)

The software block diagrams as shown in figures 4, 5, 6, 7 and 8 indicate the possible architecture of the software with the functions to be implemented. The program flowchart / state chart can be captured after the system functions are further refined.

![Software block (main system functions)](image)

**Fig. 4:** Software block (main system functions)

![Software block (main system controller) statemachine](image)

**Fig. 5:** Software block (main system controller) statemachine
Both figures (3) and (4) show the overall software architecture of the system. Figure (3) shows the main functions to be implemented and figure (4) is the main control and system modes that the system can be in. Figure (3) demonstrates that the system will be expected to sense daylight, turn lights on/off, allow user to setup various modes, and display system status to user. It can also be seen from figure (4) that when the system is running it can be in one of the three modes, being daylight sensing mode, time-triggered mode and setup mode. These modes are determined by interaction with the end-user.

3. DETAIL DESIGN (SOFTWARE)

The functions in figure 4 are detailed figures 5 to 8. At the same time it is determined which groups of functions belong to the three identified modes of operation as depicted in figure 5. The UML activity diagram is used to capture the functional flow within each state to show in which sequence the functions are invoked and conditions that they are dependent upon. Within each state the functional flow itself is iterative and are implemented as a super loop.

During the sensing mode, the system starts the loop by sensing whether its daytime or not. If it is lights are immediately turned off by the system and this status is displayed to the user. Similarly by night time, the system automatically senses that it is night and therefore turns on the lights. This status is again displayed to the user. Before the loop is completed, the end user is given the opportunity to change the mode to time-based monitoring.

In the time-based mode, the system turns on the lights on/off based on the time of day. Therefore the system starts off by reading the time of day from the system real-time clock (RTC). When the current time is equal to that one set for turn off, the lights are turned off and these stay off until that time when the time for turn on arrives. At all times system status is displayed to the user and the user is also given opportunity to change system mode.

![Software block (sensing mode functions)](image-url)

Fig.6: Software block (sensing mode functions)
Fig. 7: Software block (time based mode functions)

The setup mode is simple and allows the end user to configure the system as per system requirements, including setting up the real-time clock and selecting the mode of operation.

Fig. 8: Software block (setting up mode functions)

4. IMPLEMENTATION (CODING)

The next step after detail design is implementation, and here the software program that runs the system is coded. The Arduino system provides a very convenient and easy to use environment for developing a program, including an IDE with editor and compiler, as well as USB module for downloading a compiled program to the Arduino. The following section demonstrates, via code snippets, the ease at which software can be developed to implement the designed system using built-in Arduino functions and libraries which ease development.
1) enum STATE {SENSING, TIMEBASED, SETTINGUP};

2) int lamp = 13;
   int sensor = 12;
   int modeset = 11;
   int systemset = 10;

3) STATE mode;

4) void setCurrentTime(void);
   int isTurnOffTime(void);
   int displayMenu(void);

1) The program starts with defining the possible states of the system via the STATE enumeration. The enumeration captures the modes depicted in figure 4.

2) The Arduino board itself comprises pins which are preconfigured as analog and digital I/O pins. Since our system interfaces mainly digital modules, the software makes use of digital I/O pins from pin 10 to 13. These pins are given descriptive names via int variables for easy identification.

3) The state variable is declared.

4) Function prototypes are forward declared.

5) void setup()
   {
   // Arduino configuration

6) Serial.begin(9600);  // Serial I/O used for user interaction

7) pinMode(lamp, OUTPUT);  // The actual lamp to be controlled
   pinMode(sensor, INPUT);  // Digital day-night sensor
   pinMode(modeset, INPUT);  // Digital day-night sensor
   pinMode(systemset, INPUT);  // Digital day-night sensor

   // System configuration

8) setCurrentTime();  // Ensure to update internal Time-of-day after power on
   mode = SENSING;  // default mode is daylight sensing mode

   // Splash screen
   9) Serial.println("This is the Arduino-Based Building Light Management System");
      Serial.println("Press any key to setup system");
   }

54
10) void loop()
{
    boolean day_night; ……

11) switch(mode)
{
    case SENSING:
        day_night=digitalRead(sensor);    // sense daylight, day=HIGH, night=LOW
        if (day_night==HIGH)
        {
            digitalWrite(lamp, LOW);    // turn light off i.e daytime
        }
        else
        {
            digitalWrite(lamp, HIGH); // turn light on i.e nighttime
        }

        displayStatus();           // display system status

        mode_select=digitalRead(modeset); // does user want to change mode ?
        if(mode_select==LOW)
        {
            mode=TIMEBASED;        // Yes, new mode is TIMEBASED
        }

        sys_setup=digitalRead(systemset); // does user want to setup system
        if(mode_select==LOW)
        {
            mode=SETTINGUP;        // Yes, new mode is SETTINGUP
        }

        break;

12) case TIMEBASED:

        break;

    case SETTINGUP:

        break;

    default:
13    Serial.println("System error. Should not get here!! Please reset system");
}
5) Every Arduino program starts with a setup function when both program and system initialization is performed. It is to be noted that the setup function is already declared by the arduino framework and does not have to be declared by the programmer. The setup() function is called only once at the beginning of the program.

6) In this demo system, the user of the system is interfaced via serial communications with a display on the PC console. Serial.begin() is one of the Arduino built-in functions that allows the programmer to configure serial communications.

7) Within the setup function, various pinMode() built-in functions are used to configure the Arduino pins to be used as either input or output pins.

8) The programmer can still use their own functions to further configure the system.

9) Serial.println() is one of the built-in functions used to display a message via the serial terminal.

10) Like setup(), the loop() function is also another function automatically declared by the Arduino framework. The loop() is called repeatedly and it is where continuous processing happens, similar to the normal while (1) superloop under the main() function in a normal C/C++ program. It is to be noted at this state that within the Arduino it is not necessary to declared the main() infrastructure at it is already declared.

11) The switch statement is responsible for implementing system control as depicted in figure (4). The mode variable is able to cycle through the three states since it is an enumerated state variable, with possible values being the states of the system.

12) TIMEBASED is one of the cases of the switch statement and corresponds to the TIME TRIGGERED MODE state in figure (4).

13) The default case prints “System error” message since the system should not reach here.

VI. GENERAL SAMPLE CODE FOR A BLMS

The arduino c++ code below is based on the above software blocks design. This is the general structure of a BMLS code. The code can be altered or build on depending on the system design. Comments in the code are given to explain the different sections of the code.

```cpp
enum STATE {SENSING, TIMEBASED, SETTINGUP};

int lamp = 13;
int sensor = 12;
int modeset = 11;
int systemset = 10;

STATE mode;

// NB: These functions not implemented
```
void setCurrentTime(void);
int isTurnOffTime(void);
int displayMenu(void);

void setup()
{
  // Arduino configuration
  Serial.begin(9600);    // Serial I/O used for user interaction
  pinMode(lamp, OUTPUT);  // The actual lamp to be controlled
  pinMode(sensor, INPUT);  // Digital day-night sensor
  pinMode(modeset, INPUT);  // Digital day-night sensor
  pinMode(systemset, INPUT);  // Digital day-night sensor

  // System configuration
  setCurrentTime();    // Ensure to update internal Time-of-day after power on
  mode= SENSING;    // default mode is daylight sensing mode

  // Splash screen
  Serial.println("This is the Arduino-Based Building Light Management System");
  Serial.println("Press any key to setup system");
}

void loop()
{
  boolean day_night;
  ..

  switch(mode)
  {
  case SENSING:
    break;

  case TIMEBASED:
    break;

  case SETTINGUP:
    if (!menu_displayed)       // prevent menu from displaying continuously
      mode_select=displayMenu();

    if (mode_select==SENSING)
      mode=SENSING;
    else
      mode=TIMEBASED;
    break;
default:
    Serial.println("System error. Should not get here!! Please reset system");
    break;
}

VII. CONCLUSION

Lights in Building use a significant amount of power of the overall building energy consumption. With the power loading problem faced in Botswana and many other developing countries, Building Light Management systems can help save power.

The BMLS system presented in this paper is based on the Arduino Board. The choice of the Arduino board from other Open source microcontrollers was based mainly on the programming environment, libraries, choice of a ready to use microcontroller, low cost and future system enhancements. The Arduino has an easy to use programming environment. The Arduino uses a simplified version of C++ with libraries developed for many different applications and equipment. With the Arduino board, the digital and analog pins are ready to use with headers already broken out. Other microcontrollers such as the Raspberry PI requires a specific Circuit Board (PCB) to be designed and placed onto of the microcontroller.

This paper presented a general design of a Building Lights Management System (BLMS) using an Arduino microcontroller. The overall design is relatively low and allows the system to be enhanced in the system. The hardware design, software designs and the arduino c++ code are presented in this paper. This basis of this design can be used to develop any BLMS Arduino based system.

VIII. REFERENCES


