**CORE AUDIO BEACON PROJECT**

**Version 170929**

Contents

[1. Introduction 3](#_Toc494481871)

[1.1. Architecture Overview 3](#_Toc494481872)

[1.2. Software 4](#_Toc494481873)

[1.3. Hardware 5](#_Toc494481874)

[2. Software Installation 7](#_Toc494481875)

[2.1. Core Audio 7](#_Toc494481876)

[2.1.1. Understanding Core Audio Modification 7](#_Toc494481880)

[2.1.2. How to Use 7](#_Toc494481881)

[2.1.3. Future 8](#_Toc494481882)

[2.2. Arduino Web Editor/IDE 9](#_Toc494481883)

[2.2.1. Installing the Arduino Software (IDE) 9](#_Toc494481888)

[2.2.2. Loading Code onto Arduino 10](#_Toc494481889)

[3. Circuit Design 11](#_Toc494481890)

[3.1. Electrical Circuit Diagram 11](#_Toc494481891)

[4. Debugging 12](#_Toc494481892)

[4.1. Core Audio 12](#_Toc494481893)

[4.2. Core Audio - Arduino 13](#_Toc494481894)

[4.3. Circuit & Components 14](#_Toc494481895)

[4.4. Relay Switch with Circuit & Components 15](#_Toc494481896)

[5. Finalization 16](#_Toc494481897)

[Reference 17](#_Toc494481898)

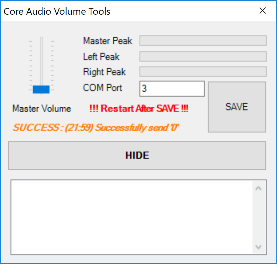
[Appendixes 17](#_Toc494481899)

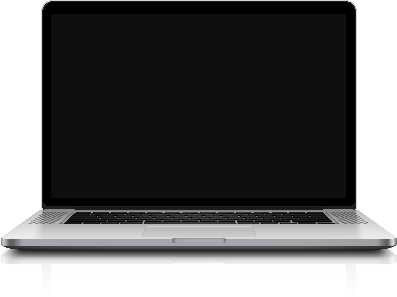
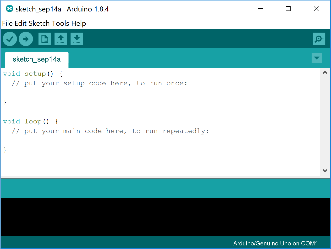
# Introduction

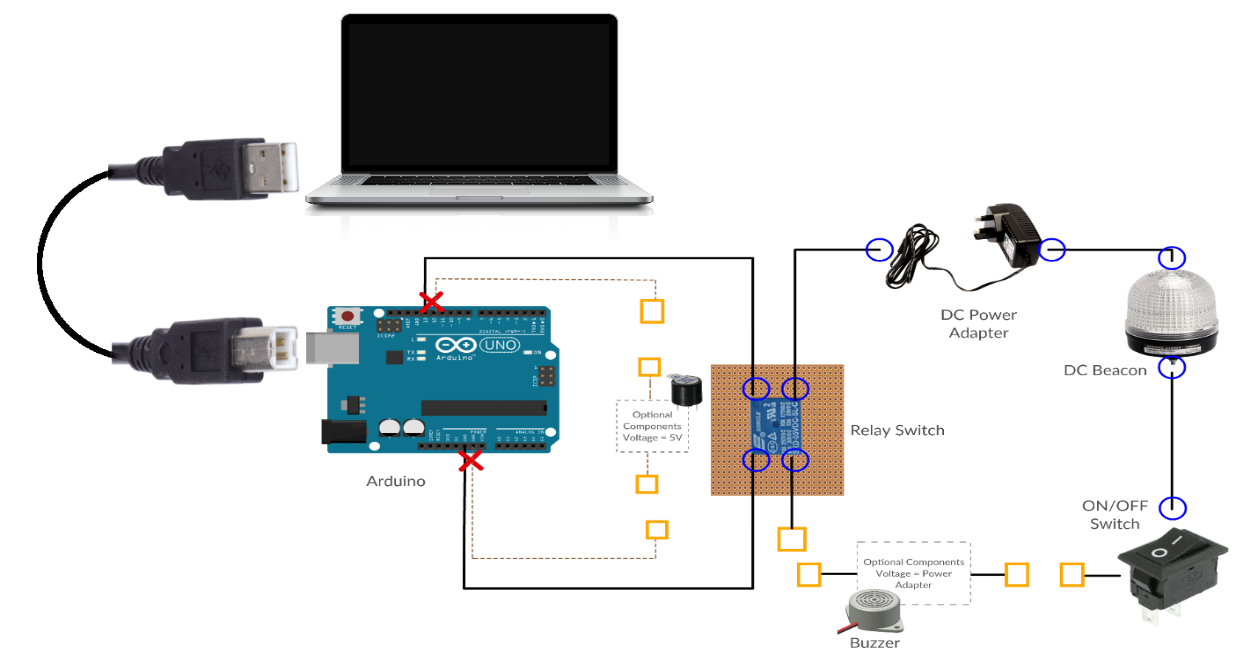
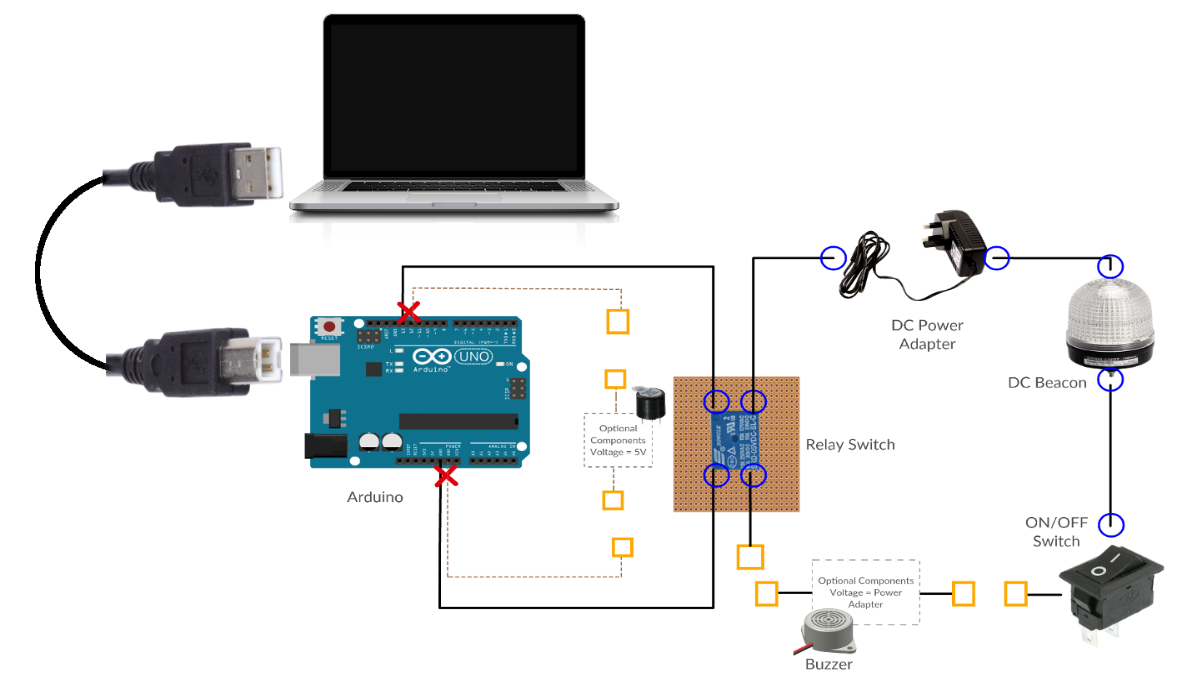
The Core Audio Beacon Project supplements the existing capabilities of TMS (Temperature Monitoring Software) by providing beacon as an alarm. The beacon can serve as an extra light and/or sound alarm in the event where TMS triggers.

The Core Audio Beacon Project builds on the alarm sound from TMS and send signal over to the microprocessor where it will activate the beacon, thereby alerting the customer.

## Architecture Overview





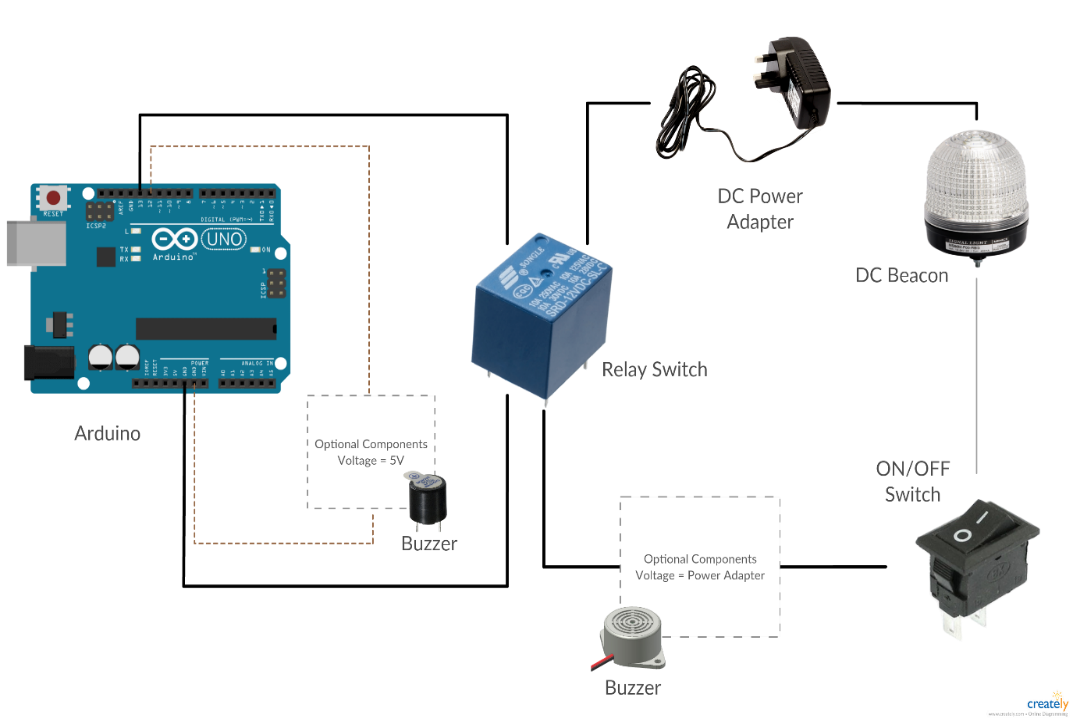


Window 7/10 PC

Installed with:

1) Core Audio

2) Arduino Web Editor/IDE



## Software

|  |  |
| --- | --- |
| Core Audio |  |
| Arduino Web Editor/IDE1 |  |

1 Refer to reference for download link. Link is working at the time of writing

## Hardware

|  |  |
| --- | --- |
| PC | Recommendation: Operating System: Window 7/Window 10 Ports: Minimum 1x USB 2.0 or 3.0 available (for connection to Arduino) |
| Arduino Uno | Image result for arduino uno |
| Relay Switch | Recommendation:  Coiled Voltage/Switch Voltage: < DC 5V Rated Load: >= (refer to Beacon specifications)  https://evola.fr/2139-thickbox_default/5v-relay-10a-250vac.jpg (sample picture) |
| Beacon (with buzzer)#  #Optional but highly recommended | Recommendation: Rated: 12VDC or 24VDC **(Do not get VAC/AC)**  Buzzer: Built-in  Image result for 24vdc beacon  (sample picture) |

|  |  |
| --- | --- |
| DC Voltage Power Adapter | Recommendation: Input: **Within standard voltage in Singapore**  Output: (refer to Beacon specifications)  Image result for 24v DC power adapter |
| Buzzer#  #For beacon with no built-in buzzer | 1) If placement of buzzer is with beacon, get the same voltage rating as the beacon[A] 2) If placement of buzzer is with Arduino, get a 5V buzzer[B]  [A] Image result for buzzer [B] Image result for buzzer |
| Wire & Breadboard | For debugging before actual soldering  https://www.allaboutcircuits.com/uploads/articles/breadboard_jumpers.png Image result for breadboard |
| PCB & Soldering Iron & Lead | Image result for pcb boardImage result for soldering ironImage result for soldering lead |
| On/Off Switch (SPST) | Image result for SPST |

# Software Installation

## Core Audio

Core Audio is a C# program using Windows Core Audio APIs. The core audio APIs provide the means for audio applications to access audio endpoint devices such as headphones and microphones. The core audio APIs serve as the foundation for higher-level audio APIs such as Microsoft DirectSound and the Windows multimedia waveXxx functions. Most applications communicate with the higher-level APIs, but some applications with special requirements might need to communicate directly with the core audio APIs.

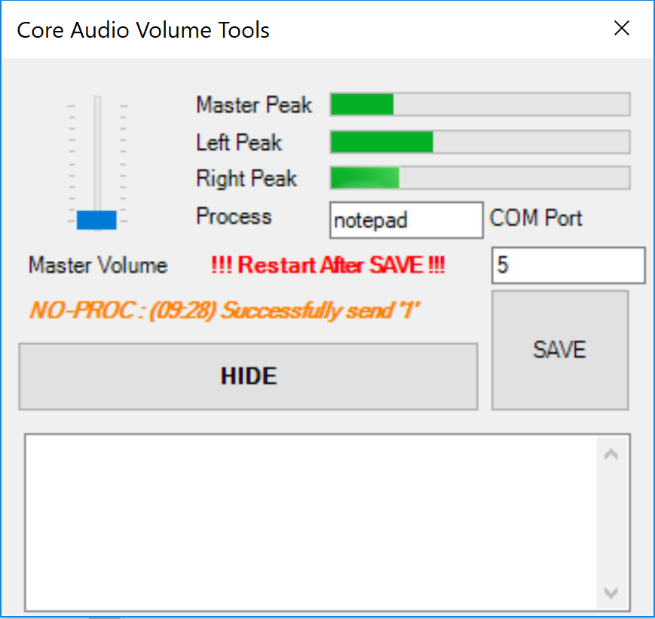


### Understanding Core Audio Modification

The original core audio sample program can be downloaded at codeproject3. The original project has been tested to work on Window 7 and Window 10.

To original program was modified to include 3 functions:

* A simple user interface to allow user to modify the COM port number and process name.
* Communication to COM port, so that message can be send out.
* Process checking, to check whether process is currently running.



**5**

**6**

**1**

**3**

**4**

**2**

### How to Use

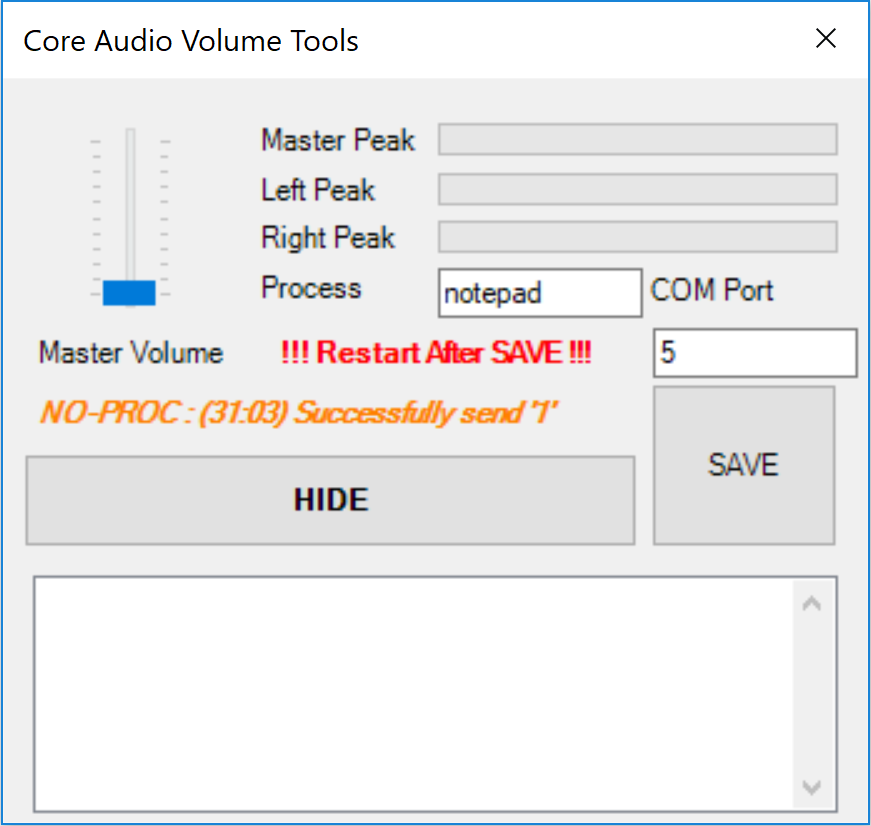
1. This is the volume mixer. When sound is detected on the system, green bar will fall and raise.
2. Input the process to be check. It is important to enter the process name without .exe extension. Go under Task Manager > Details to find the name of the process. (Example *notepad.exe = notepad, chrome.exe = chrome, tms.exe = tms*)
3. Input the COM port no. of where the Arduino is connected. COM port no. can be found at Device Manager.
4. This shows the status information of Core Audio. There are 3 types of status;
5. *“ERROR”*: indicates that there is problem establishing connection with the COM port



1. *“SUCCESS”*: indicates COM port connection is establish and message is send



1. *“NO-PROC”*: indicates that processes could not be found



1. Hide minimize Core Audio to tray
2. Save button updates the new values of COM Port and Process. User must RESTART for new values to take effect.

### Future

For future extension, the Core Audio program can be replaced with any program that has COM port function. The replacement should adhere or take into consideration the guideline to prevent compatibility issue with Arduino.

1. Program should only send in ASCII format
2. To sound the alarm, program should send ASCII ‘1’
3. To turn off alarm, program should send ASCII ‘0’
4. Alarm will always sound when serial availability is lost

## Arduino Web Editor/IDE

Arduino Web Editor/IDE is needed to interface with the Arduino hardware. User can choose to use the Web Editor or IDE to load code onto the Arduino.

Get the latest version from the Arduino download page4. You can choose between the Installer (.exe) and the Zip packages. It is suggested to use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.

To learn more about Arduino, refer to their official website1.

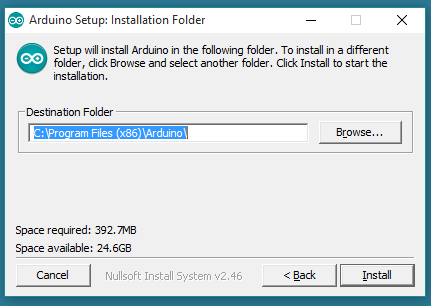


### Installing the Arduino Software (IDE)

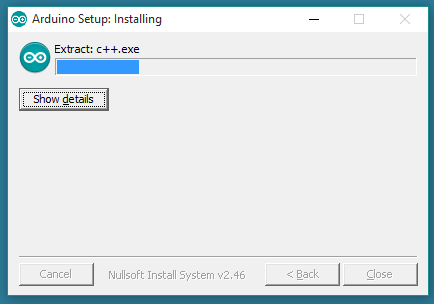
1) When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



2) Choose the components to install



3) Choose the installation directory (user default one)



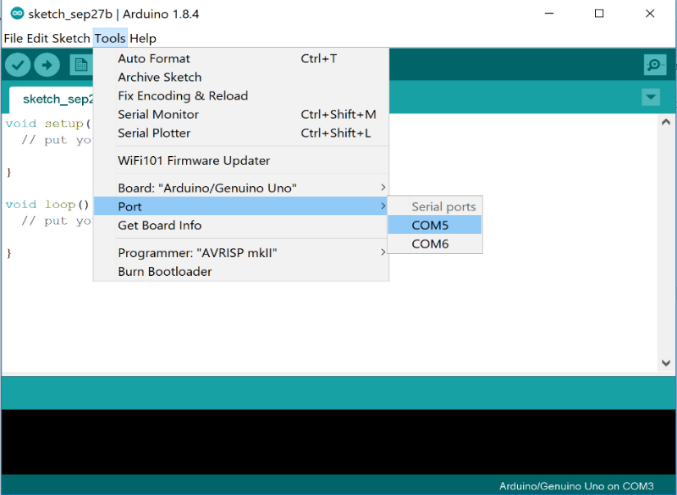
The process will extract and install all the required files to execute properly the Arduino Software (IDE)

### Loading Code onto Arduino

1) Connect Arduino to PC > Start Arduino IDE

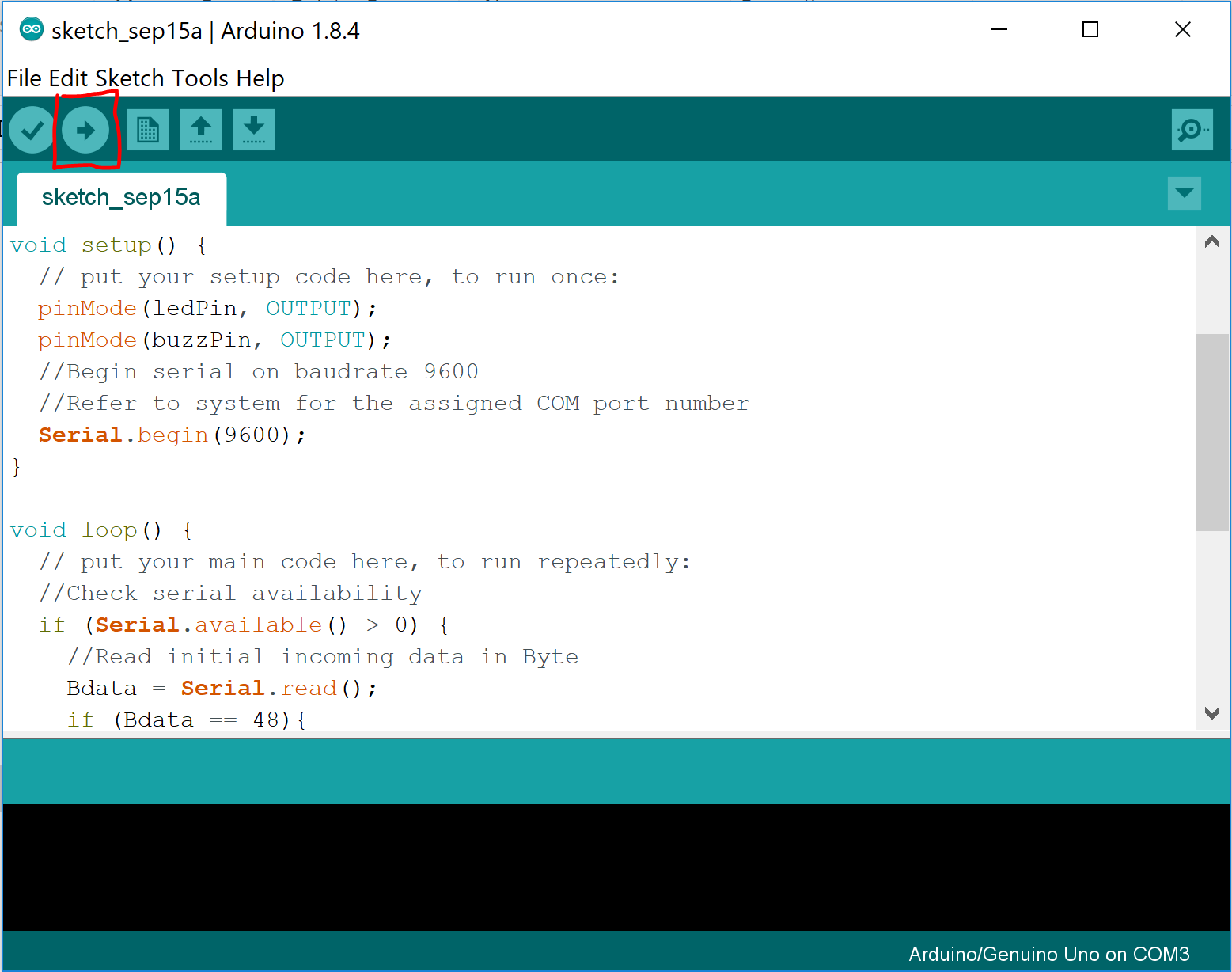
2) Select Tools > Port > *COM port no. of the Arduino*#

#COM port no. can be found at Device Manager



3) Copy code into the editor, code can be found at *Appendix A*.

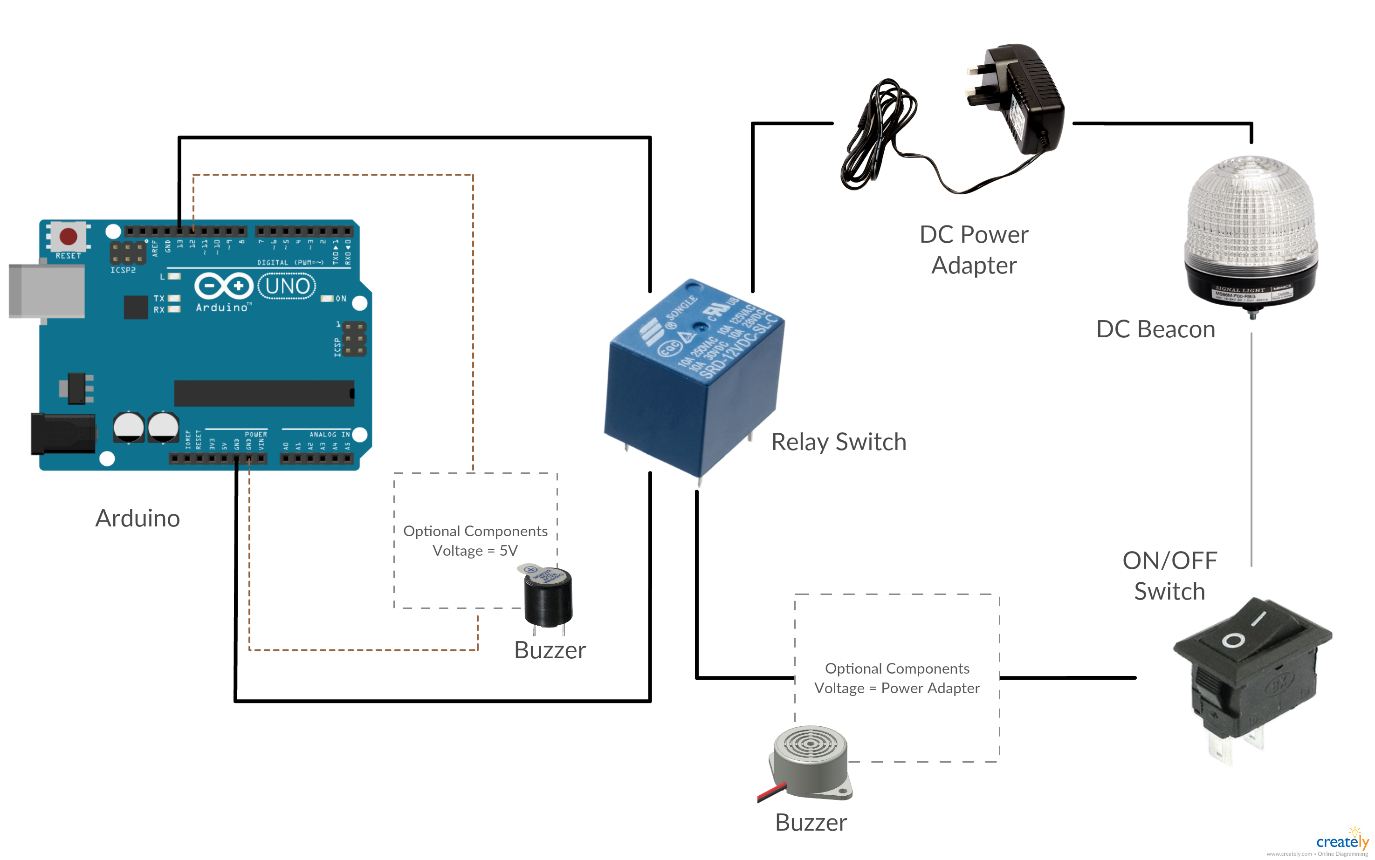
4) Upload the code onto Arduino by clicking the right arrow



# Circuit Design

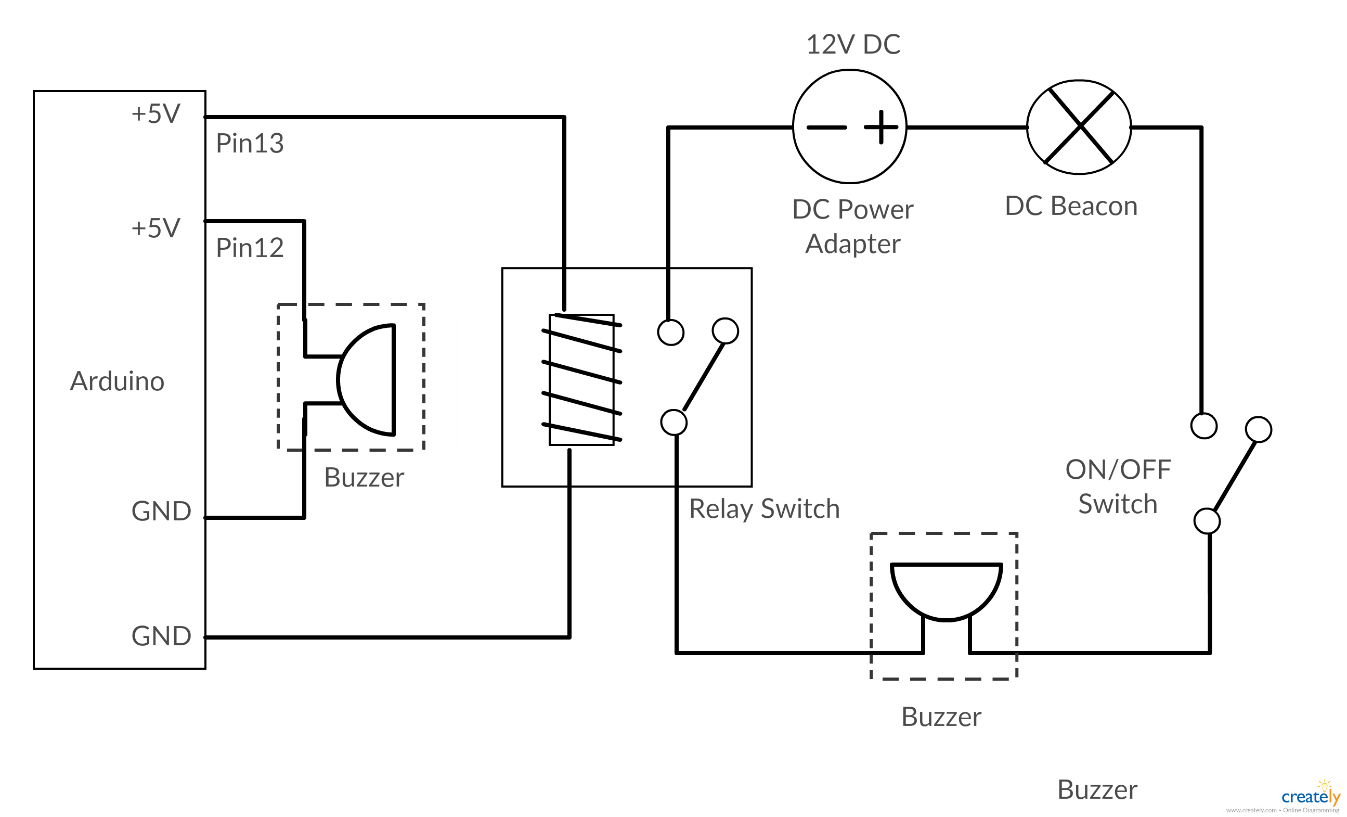
## Electrical Circuit Diagram

* All the following diagram is design based on 12V specification. #
* Circuit Design should be done on breadboard so that configuration can be change easily.



**12V**

**5V**



**5V**

**12V**

#The beacon can be swapped with a higher voltage, however, ensure matching voltage &  
compatibility for the output from power adapter, rated load for relay switch, buzzer and on/off switch

# Debugging

## Core Audio

There are 3 types of status;

1. *“ERROR”*: indicates that there is problem establishing connection with the COM port



* Possible Problem:

COM port conflict, other program is using the same COM port number

Resolution:

1. Change the COM port number used by Arduino under Device Manager
2. Update Core Audio with new COM number and restart
3. Restart PC

* Possible Problem:

COM port does not exist

Resolution:

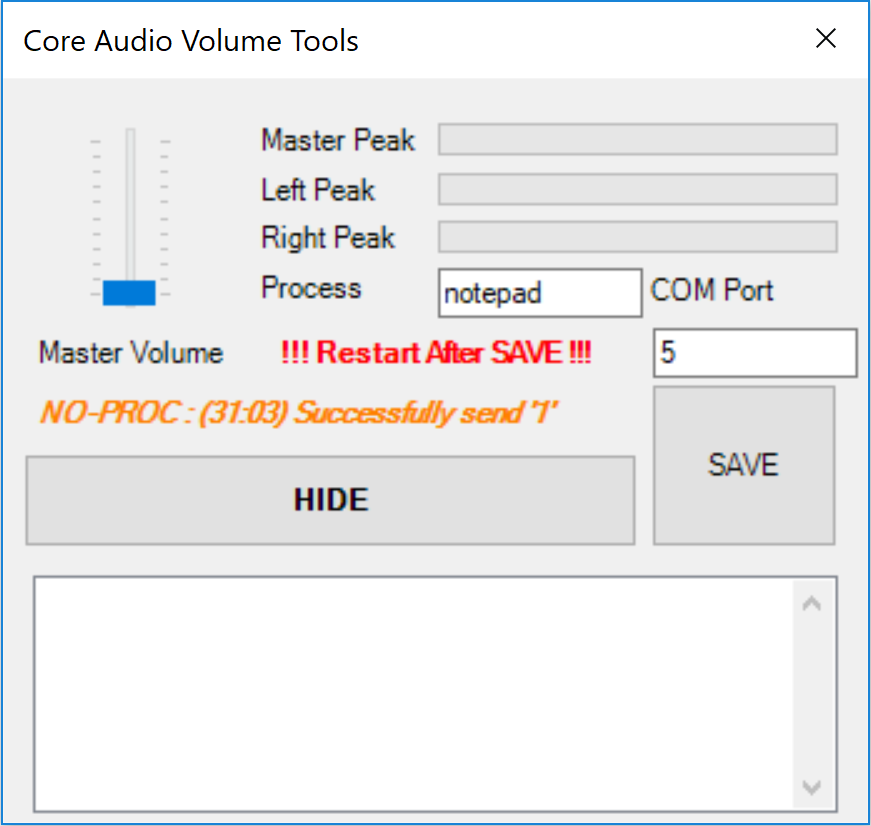
1. Check COM port under Device Manager
2. Restart PC when COM port exist

* Possible Problem:

Setting did not take effect of the new value

Resolution:

1. Ensure setting is correct
2. Restart Core Audio
3. *“NO-PROC”*: indicates that processes could not be found



* Possible Problem:

Process does not exist

Resolution:

1. Go to Task Manager
2. Ensure process exists

* Possible Problem:

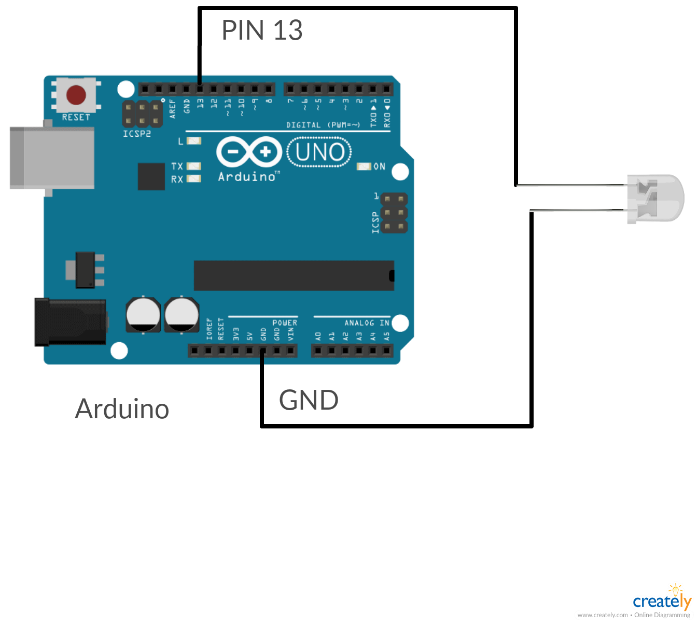
Process name is wrong / Process name contain .exe

Resolution:

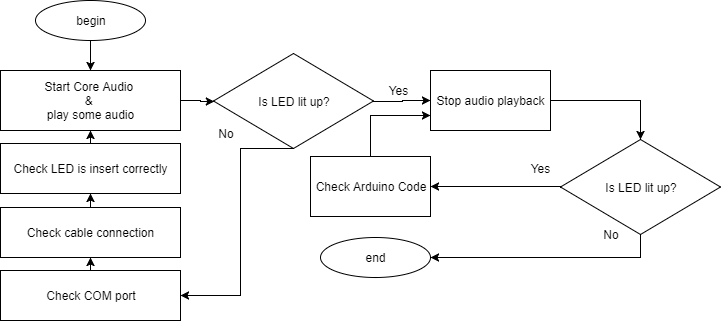
1. Go to Task Manager
2. Ensure process exists
3. Enter again the process name with *“.exe”*
4. Restart Core Audio

## Core Audio - Arduino

1. Find the COM port of the connected Arduino
2. Ensure code is loaded onto the Arduino (refer to *appendix A*)
3. Insert LED to PIN 13 and GND

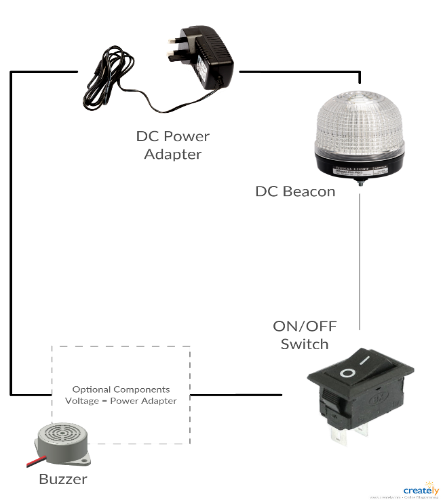


1. Debug using the flowchart

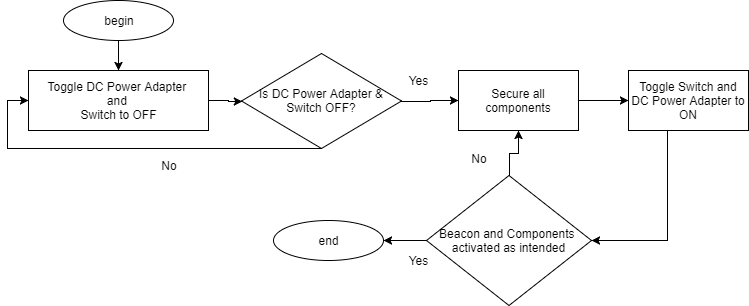


## Circuit & Components

1. Setup circuit as shown below

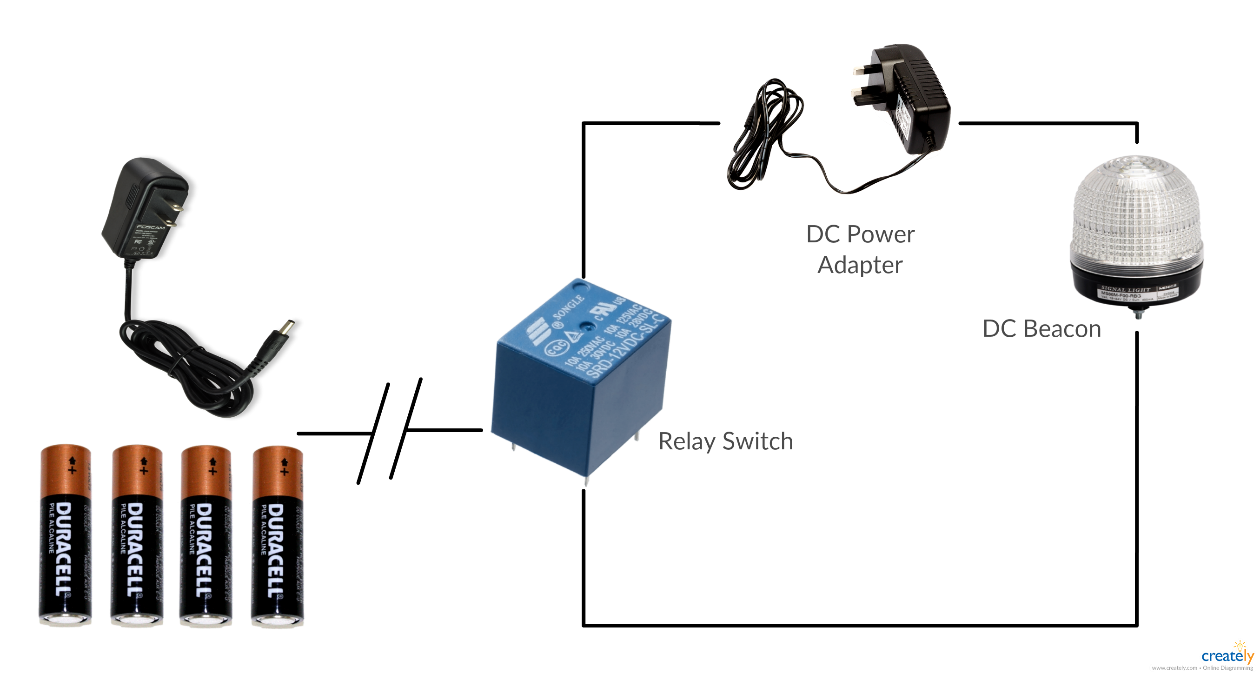


1. Debug using the flowchart

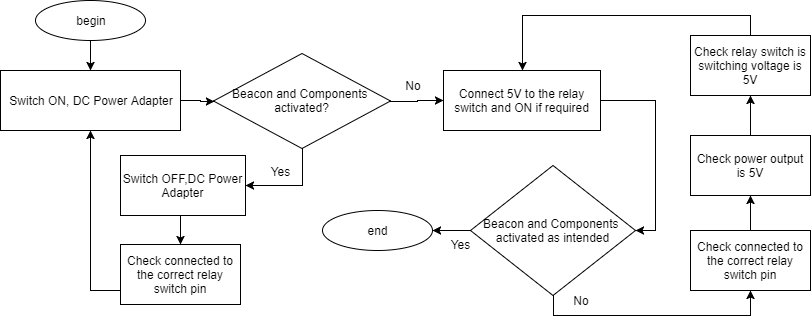


## Relay Switch with Circuit & Components

1. Setup circuit as shown below
2. Prepare 5V DC Power Adapter or any other alternative 5V Output
3. DO NOT CONNECT the 5V to the relay switch yet



1. Debug using the flowchart



# Finalization

Checklist:

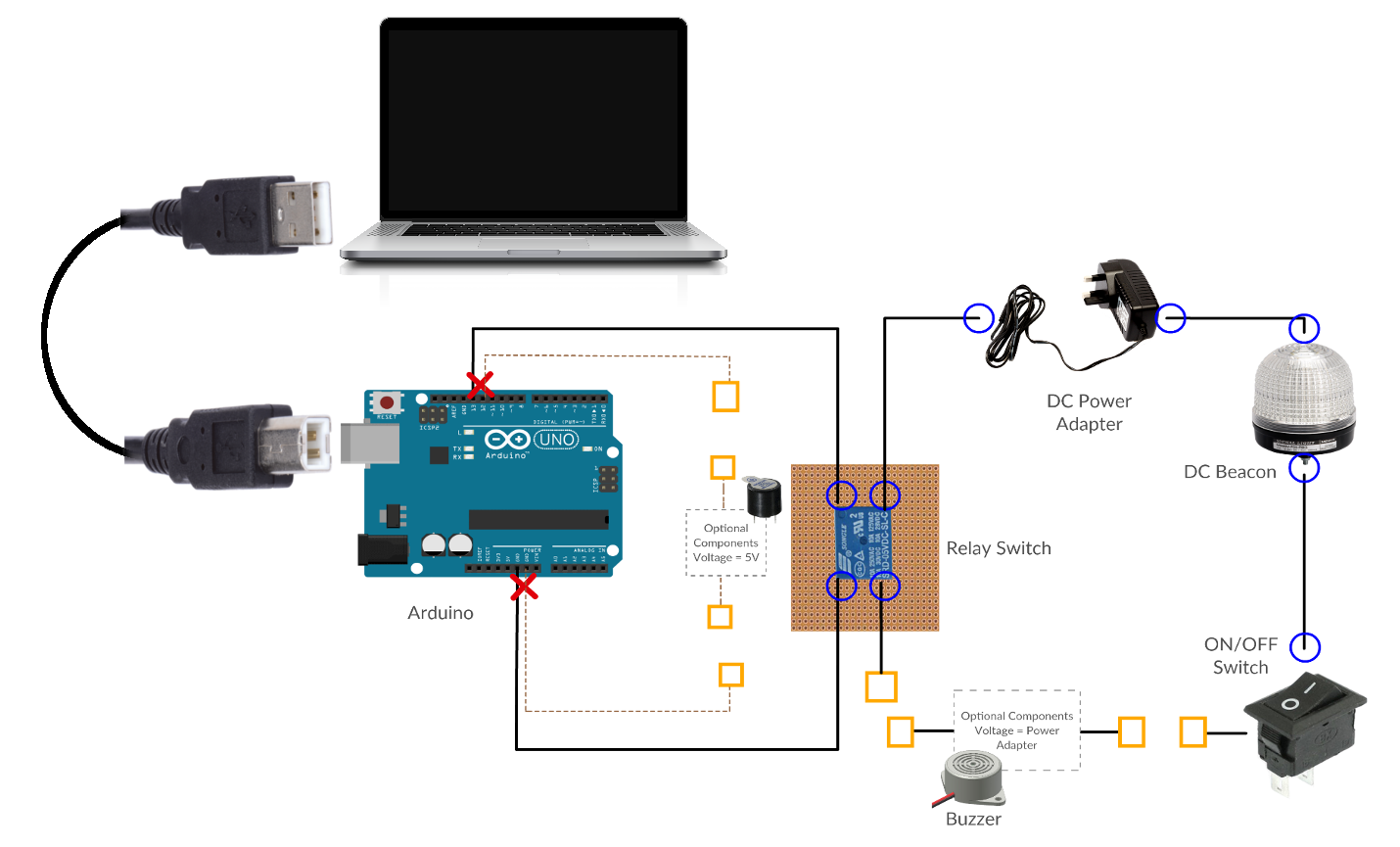
* Ensure hardware configuration design is finalized
* Ensure software and code is working
* Debugging is done and components are working as intended

Preparation:

* Soldering Iron and Lead is available
* PCB is large enough to accommodate all wiring pin and relay switch onboard

Soldering:

* Solder based on the diagram below:
  + Red Cross: Do not solder
  + Blue Circle: Solder
  + Orange Square: Use a wire connector



# Reference

1 https://www.arduino.cc/

3 https://www.codeproject.com/script/Articles/ArticleVersion.aspx?aid=18520&av=112029

4 https://www.arduino.cc/en/Guide/Windows

# Appendixes

#### Appendix A

//PIN for trigger

int ledPin = 13;

int buzzPin = 12;

//for incoming byte data

byte Bdata;

void setup() {

// put your setup code here, to run once:

pinMode(ledPin, OUTPUT);

pinMode(buzzPin, OUTPUT);

//Begin serial on baudrate 9600

//Refer to system for the assigned COM port number

Serial.begin(9600);

}

void loop() {

// put your main code here, to run repeatedly:

//Check serial availability

if (Serial.available() > 0) {

//Read initial incoming data in Byte

Bdata = Serial.read();

if (Bdata == 48){

digitalWrite(ledPin, 0);

digitalWrite(buzzPin, 0);

}else if (Bdata == 49){

digitalWrite(ledPin, 1);

digitalWrite(buzzPin, 1);

}

}else{

digitalWrite(ledPin, 1);

digitalWrite(buzzPin, 1);

}

delay(500);

}