**Escape Room Season 1: System Design Document**

# Objective

This document provides information on the Escape Room Season 1 project, which will be called **Escape Room 1** and **System** in this document. This document was created with long-term storage of the project between uses and maintenance of the System. This document is subject to change due to modifications to the Escape Room 1 project.

NOTE: This document does not discuss puzzle ideas or reasoning behind the final implemented puzzles.

The parts of Escape Room 1 this document discusses include hierarchy of system devices, languages used in programming the system and coding practices thus, setup of system devices, maintenance of the System for general use and when modifying system devices, wiring of the System, and budget of the Escape Room 1 project.

## Background

This project’s main objective was to create an escape room using a large amount of technology not usually found in escape rooms, which is why this section will briefly describe the basic principles and general setup/goal for an escape room.

The basic principle of an escape room is that it is an activity where an individual or a group of individuals are placed within a space which is usually a room within a building or an entire building depending on the escape room scope that tends to be filled with puzzles and a countdown timer. The escape room activity usually ends when the timer runs out of time or when the individual(s) solve all the puzzles within the escape room space.

The general setup of an escape room is a room that is filled with props to setup the story environment, and props to use to solve puzzles located within the room. There tends to be smaller puzzles (**sub-puzzles**) that are subsections of one larger puzzle, which is the overall goal of the escape room. All puzzles are planned out by the escape room designer and are usually designed to take an average person the full time on the timer to solve all of the puzzles. The difficulty also depends on the escape room designer; the previous sentence is the most often setup preferred by escape room designers.

# System Overview

Escape Room 1 deals with a variety of devices that can be viewed in the Budget section of this document. The overall hierarchy structure of the System can be viewed in Figure 1 and begins with a Raspberry Pi 2 model A (a.k.a. Pi) which contains the logic for the System’s puzzles. This primarily deals with both device to device, and puzzle to puzzle transitions. In general terms, the Pi coordinates device interactions within a smaller sub-puzzle in order to manage and complete the sub-puzzle as well as coordinates transitions between sub-puzzles in order to complete the larger puzzle. The Pi tends to do this by sending strings containing command codes via Bluetooth to various micro-controllers. In this project, we use 4x Arduino Mega 2560s rev2 with HC-05 Bluetooth modules which are capable of both master and slave modes as our micro-controller of choice. The Pi also manages a keyboard and a display for output, which can be used for either admin use or for Escape Room 1 puzzles.

As mentioned in the previous paragraph, there are 4 Arduinos used in the System for the primary use of managing devices necessary for the functionality of Escape Room 1. These are generally then grouped into locations where the devices will be used and same devices are grouped together as much as possible. There are 2 Arduinos located within a Box container and for easier reading, the first Arduino will be called the NFC Arduino and the second Arduino will be called the Main Cube Arduino. There are also 2 Arduinos that handle other devices elsewhere in the room and these will be called the Magnet Arduino and the Laser Arduino. These Arduinos manage the majority of devices needed within Escape Room 1, which include but is not limited to lasers, NFC reader, solar cells for sensors, and a keypad. For a more in depth list, please refer to the Budget section.

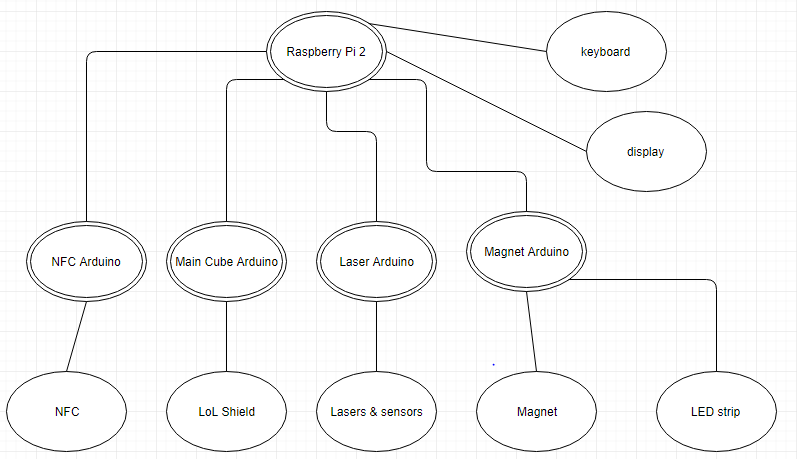


Figure 1: Device Structure of Escape Room 1.

# Programming the System

There are 2 main programming languages used within the System. Python was used to program the Pi because there was a large pool of python examples provided online by the Raspberry Pi & Python community and it provided easy implementation for various puzzle options that were thought of in the System Design stage, but are not implemented in the final product. The programming in the Python script includes Bluetooth communication between the Pi and the various Arduinos, uni-curses for display using Python, and the logic for sub-puzzle/escape room completion.

To program the Arduinos, the Arduino library and language was used which contains a c++ similar language. The NFC Arduino handles all transmitting CAT5 puzzle pieces, PN532 (Arduino capable NFC reader/writer), 12 LED ring, and the cube inner light. The Main Cube Arduino handles displaying the necessary lit lights on the LoL shield, keypad membrane, and receiving all CAT5 puzzle pieces. The Magnet Arduino handles an electromagnet, LED strip for lighting within the room, and 5 servo motors. The last Arduino, the Laser Arduino, manages the sensors, lasers, small LED diodes, and a PanTilt servo motor. The specific programs can be found in the Appendix of the document or in the Escape Room code repository/google drive.

## Coding Practices

When the use of a symbol is described in this section, the symbol will be shown in parenthesis.

### Naming Conventions

For both the Arduino code and the Python code used for the Raspberry Pi, lowerCamelCase naming convention is used for all variables and function/method names not created by a ‘#define’ method. UpperCamelCase is used for all variables defined by a ‘#define’ method and all classes/objects.

### Function Management

All function blocks of code start with an opening brace ( { ) and end with a closing brace ( } ). In addition, all functions and global variables that correspond to a specific device or functionality can be found in sub sections at the beginning of the Arduino program (.ino) files, where each sub section begins with a C++ multi-line comment which describes the functionality and/or device the sub section is used for and the sub section ends with a C++ multi-line comment which explicitly says it is the end and re-iterates what functionality and/or device the sub section dealt with.

# Setup of System Devices

This section describes the setup of all devices for the System which includes how the placement of devices was done, and what testing was done to various electronic devices in order to find the necessary working conditions which can be found in the finished circuit schematics that can be found in the Maintenance section.

## Testing of Device Specifications

This section describes what devices we verified the working specifications for and what the working specifications are. We had to check what the working specifications were because the majority of the specifications found in datasheets are maximum ratings and actual ratings vary between devices.

We tested the 5mW red lasers which had maximum ratings (maximum working specification) at 5V when working with a current of 25mA but when testing we found that it works at 5V when working at a current of 20mA. We also tested 2 kinds of solar cells to use as sensors for the lasers, where we got the first kind of solar cells from the local Princess Auto store in the Surplus section. These solar cells were too fragile and broke often, and did not have any wires with a solder connection on them to allow for easy wiring. In addition to the second kind of solar cells which were found on the Digikey website, that cost approximately the same amount of money (not including shipping cost) but had solder connections to wires for easy wiring and an online datasheet.

We also tested the NeoPixel ring which was found on the Adafruit website by soldering on wires for the necessary connections and running various example programs which were given with the installation of the Adafruit NeoPixel library. With the use of the library, we found the desired brightness we want the ring to be was 16% brightness (40 out of 255-full brightness) and we found that the type of the ring setup which we bought was NEO\_GRBW which was important because it is needed in the creation of the Adafruit NeoPixel object in the Arduino programming code.

We also tested the **LoL** (Lots of LEDs) **shield** which we borrowed from a Professor at the University of Regina called David Gerhard, by using example programs found online after some research. During this research we found out that the unprotected backs of the soldered connections may touch the exposed metal on the Arduino which will cause the LoL shield to act unpredictably when the Arduino is running the example programs.

The last device we tested to ensure we understood how it works and how to program it was the NFC reader/writer which is compatible with an Arduino. We found the NFC reader on the Adafruit website which is called the **PN532**. Here we tested how far away it could detect various NFC cards and tags, and we found that we can scan the bus passes used by Regina Transit the farthest amongst all the testing NFC cards and tags. We tested basic NFC tags (Mifare Ultralight) which were found on the Amazon website which had to have direct contact with the PN532, we tested the Mifare Classic NFC card which was given with the PN532 and it was detected at a maximum distance of 1cm, and we tested Mifare Classic NFC card which I used over the years as a Regina bus pass (which is now retired as a bus pass) and it had a maximum distance of 4cm. For the testing, all cards were held so the length of the NFC card and the length of the reader were held parallel to each other which is the optimal positioning of the card to get detected. We found this optimal positioning information off the Adafruit NFC information page (L., 2012) and from personal experimentation.

We also tested the Bluetooth communication between devices in order to test that communication between devices was possible for the System. Unfortunately we only tested the use of the Bluetooth communication in the situation where there was constant communication between one Arduino and one Raspberry Pi, which we realised later on in the project that this was not always the expected usage for the communication.

## Testing Placement of Devices

For various devices we used temporary circuits in order to determine the placement we want the devices to be within the room. For the rest of the devices used in the System, we chose locations in the room for the devices where the devices and device enclosures would make the room seem smaller and more esthetically pleasing to the Escape Room planner, Oles Shnurovskyy.

For the placement of the lasers, we connected one laser to one 3V CR2025 watch battery and repeated this as many times as we had loose lasers (lasers not located within an enclosure) to place within the room.

We also tested out the placement of 10mm LED diodes/lights individually by attaching one 3V CR2025 watch battery to them and we tested the 5mm LED diodes/lights individually by attaching a small circuit which ensured a current of 20mA at an assumed voltage of 3V for the diode/light.

# Maintenance

This section deals with the maintenance of the System, which includes how the System was first created, and what steps should be made in order to do changes easily. In order to ensure ease of maintainability, this section contains circuit schematics of the System which are each divided into sections based on electronic device. This section also describes what devices/functionalities each Arduino within the System controls/manages and describes the logic of the programming code that corresponds to each device/functionality.

In addition, this section also briefly describes the programming code meant for testing each device/functionality in order to ensure ease of testing or modification of tests after a device or functionality is modified.

NOTE: You can find the programming code that corresponds to each device at the top of the associated Arduino programming (.ino) file.

### Circuit Schematics

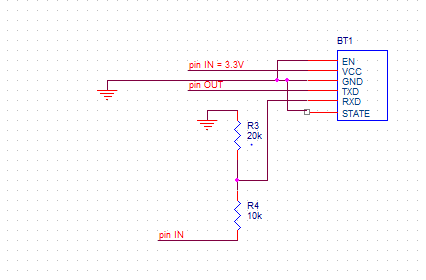


Figure 2: Circuit used for Bluetooth Module. Each Arduino has one.

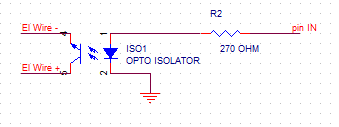


Figure 3: The Arduino tells when the switch (opto-isolator) turns on which then turns on the AC glowing EL Wire.

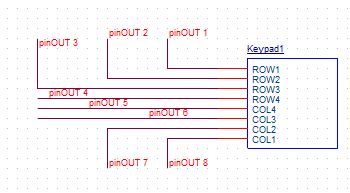


Figure 4: The planned connection between the keypad membrane to the NFC Arduino.

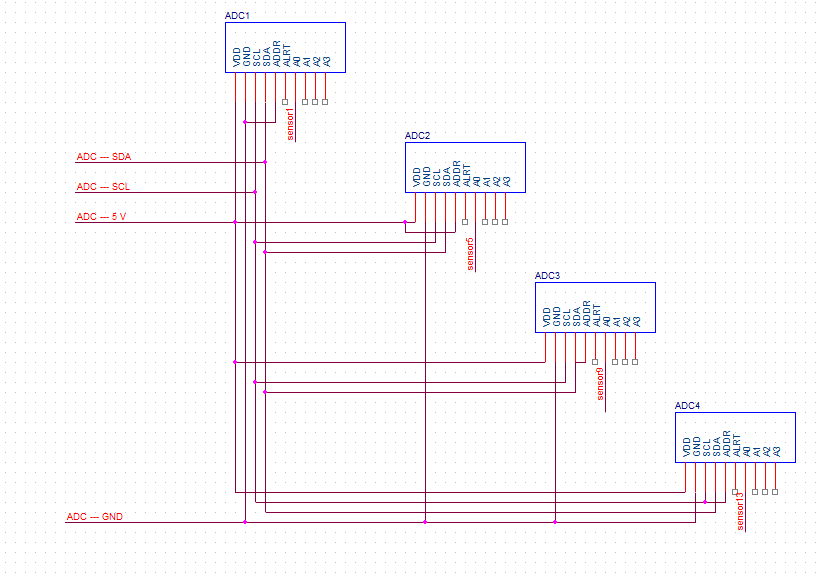


Figure 5: The wiring of the several ADCs needed for the System to one Arduino, the laser Arduino.

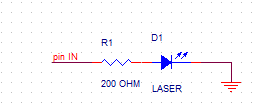


Figure 6: The final circuitry decided on to ensure the long life of the laser.

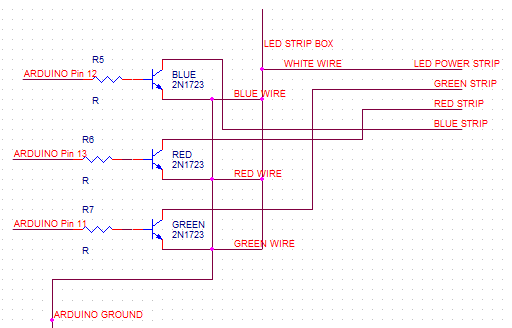


Figure 7: The circuit to ensure the programmatic use of the LED light strip where an analog signal can be used.

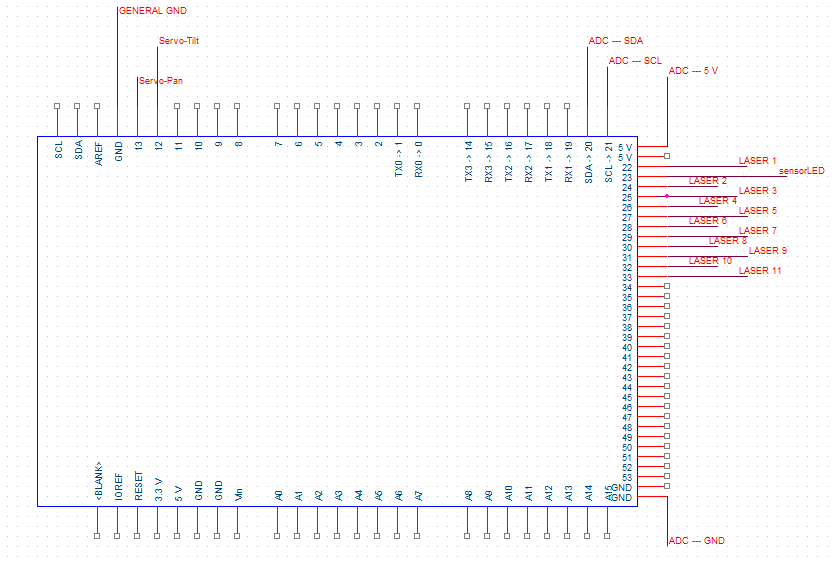


Figure 8: The layout of the pins and what uses them on the laser Arduino.

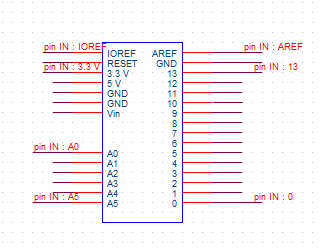


Figure 9: The pins the LoL shield uses. NOTE: All pins are used if used with an Arduino Uno.

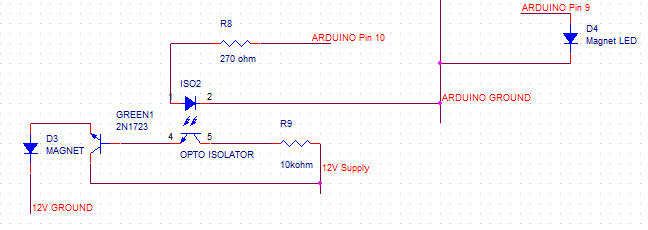


Figure 10: Circuit for both the door Magnet and the magnet arming LED.

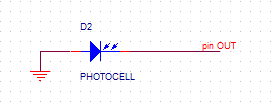


Figure 11: Circuit used for the solar cells when they are used as sensors. NOTE: ‘pin out’ is a wire which is then connected to an ADN channel which is then converted to a digital number for processing by the laser Arduino.

### Devices and Their Placement in Arduino Files

## Maintenance for Adding/Removing Devices

## Testing

# Wiring and Enclosures

# Budget

# Conclusion

Any maintenance for Escape Room 1 should be do-able with reference to this document, and fair amount of information should be given to reproduce this project at any given time with respect to devices used and functionality of devices. Feel free to modify the length of wire used and enclosure for devices used.

# References

L. (2012, December 30). Adafruit PN532 RFID/NFC Breakout and Shield. Retrieved October 01, 2017, from https://learn.adafruit.com/adafruit-pn532-rfid-nfc/mifare