ENIGM-DESK
PROGRAMMER GUIDE
V1.0

Client :
GONZAGUE YERNAUX
Project manager :
AUGUSTIN NOGUÉ

2020-2021

Contents

1 Introduction

This **programmer guide** describes the different tools and process used to create this project. **ENIGM-DESK** is an escape game composed of a digital photo frame which include a Raspberry Pi 4 model B and a Arduino Mega 2560, an RFID reader MFRC522 and a 7 inch touch screen. Outside the digital photo frame there is a load-cell connected to HX711, a "precision 24-bit analog to-digital converter (ADC) designed for weigh scale" and a solenoid that can be setup on any desk with a drawer. Raspberry Pi 4 model B.

We will start by explaining all the different part needed on the Raspberry Pi and then on the Arduino.

This project was realised for a class at the **Unamur** university and alongside my client / teacher **Monsieur Gonzague Yernaux** who I need to tank for his help and all the nice conversations we shared while designing this Escape game.

2 Components

You can find all the information link to the components in the User Guide

3 Wiring

In order for the code to run on your Raspberry Pi you will need to follow this precise wiring. You can find the full Fritzing code on this project's GitHub.

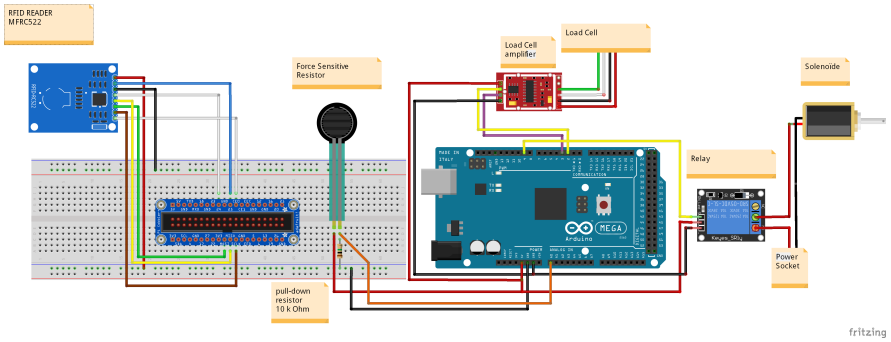

Figure 1: Full electronic circuit done with Fritzing.

4 Code layout

The code for this project is divided in two big parts, the code running on the Raspberry Pi and the one running on the Arduino Mega 2560. For the Raspberry Pi the code is divided in four parts, the main.py file with the GUI, the raspberry.py with all the functions for the communication with the Arduino, accessing files, mounting and unmounting of the floppy drive. The test.py for the unit testing and finally the configuration.py with all the configuration that can be modified for future use such as the speed of the digital frame or the mounting point for the floppy drive.

For the Arduino Mega 2560, all the code is located in the arduino.ino in the arduino directory.

5 Raspberry Pi

5.1 Introduction

There is only three things connected to the Raspberry pi, the Arduino, the floppy drive and the RFID Scanner but the main part of the project is running on it, the GUI made with Tkinter.

5.2 RFID

In order to make the RFID reader MFRC522 work on a Raspberry pi you will need to activate the SPI protocol. To do so you will need to input the following command inside the Raspberry Pi terminal :

```
sudo raspi-config
```

After that you will need to go in "Interfacing Options" and then "SPI" and choose 'YES"

For the rest of the RFID reader MFRC522 the project is using **Ondřej Ondryáš's** library, Python RC522 library.

5.3 Connection with Arduino

5.3.1 Configuration

To make it as simple as possible the two main parts of the projected to each other via and USB cable. This way there isn't any big configuration needed it's almost a "plug and play" solution. You just need to use the "serial" module:

```
import serial  # Module for communication between the arduino and the Raspberry Pi
```

You will be able to find the only configuration needed inside the configuration.py file located in the game folder :

```
ser = serial.Serial("/dev/ttyUSB0", 9600, timeout=10)
```

Be sure to use a timeout to let the connection between the Arduino Mega 2560 and Raspberry Pi to establish itself or it will cause issues and make it unstable.

5.3.2 Communication

In order to pass communication between the Arduino Mega 2560 and the Raspberry Pi, we are using strings converted to bytes that are then pass through the serial communication. We have elected to use a command system coming from the Raspberry Pi to the Arduino Mega 2560, this way the Raspberry Pi is requesting actions or update to the Arduino Mega 2560 that stays passive.

Which means only four functions are necessary on the Raspberry Pi side of it :

- setup arduino
- cleanup arduino
- deactivate solenoid
- get arduino sensor status that pass as an argument the name of the sensor and request for an update than will be returned as a string. In order to read messages coming from the Arduino the function needs to decode it first.

5.4 GUI: TKinter

All the utilisation of Tkinter made is in this project is based on the book *Tkinter Gui Programming By Example* by **David Love**.

The main aspect of this project revolves around using clickable zones instead of buttons, are it's more flexible and often more aesthetic.

In this project you have five clickable zone that are activated by outside events. Those five zones are :

- Top left corner to go back but to the menu and if you haven't won
- Top right there is an small hidden zone to leave the game
- Middle right and middle left to execute actions
- Bottom of the screen once you have passed all the victory steps

All of those zones are managed inside a function that records left clicks or taps on the screen and will take actions depending on the coordinates of that record.

The rest of the functionalities are explained in a much professional way and in great depth in *Tkinter Gui Programming By Example* by **David Love**.

5.5 Floppy disk

The floppy disk installation and usage is frankly based on wizardry more than logic, but it seems like there is a work around. At the first the idea was to automatically mount and "unmount" the floppy drive like on windows were when you try to access a directory the drive is mounted automatically. Unfortunately on Linux it's not the case. So in order to do so you need to pass commands to the Linux's terminal. That's also how i discovered

that It's possible to pass "sudo" commands from a python script without it being called by a sudo command itself.

So all the system behind is revolving around passing "sudo" commands to mount and unmount the floppy drive every time it's needed with the command :

```
mount_command = "sudo mount -t msdos /dev/sda /media/pi"
os.system(mount_command)
unmount_command = "sudo umount -t msdos /dev/sda"
os.system(unmount_command)
```

6 Arduino

For the Arduino the code is even more straight forward, the setup initialize the serial communication with the Raspberry Pi and the main loop catches and transform the different communications in order to respond to each request correctly. In the main loop you will also find the update functions for the Force resistor sensor and the Load cell.

The code for the serial communication is based on the example available directly with the Arduino and made by Tom Igoe and the one for the load cell buy the maker of the HX711 by Nathan Seidle