Project Report For New Devices Lab

Playing Card Prediction Using IR

Luna-Elise Schernthaner (s4703928) Yannick Hogewind (s4413172)
Nick van Oers (s1009378) Jordy Aaldering (s1004292)

Abstract

Our project for the New Devices Lab is to predict playing cards on a stack. We built an algorithm to predict the cards based on markings on the sides and translate this to a buzzing device that lets its wearer know which cards are next on the stack.
This report describes the architecture of the code. A user manual is also provided with hardware requirements. The last part of the project contains a description of the demonstration video.

Contents

1 Introduction

Figure 1: A stack of cards with the markings that encode every playing card

The goal of the project was to predict playing cards on a stack by means of invisible markings on the sides using infrared ink. During the project, it became clear that we would just have to work with human-visible ink, as the infrared ink would have brought many complications.
The encoding exists of two small markings, to let the algorithm know where the markings start, followed by four markings that can be full, left or right (see Figure 1). This means $3^4 = 81$ different cards could be encoded, but this project only required 52 different cards, therefore some codes do not correspond to a card.

These are the steps of the algorithm:

1. First the deck of cards and the area with markings are detected (see Figure 2) and the image is cropped so the markings are as horizontal as possible and there is nothing else inside of the cropped image
2. It then translates the encodings to the corresponding cards
3. Then the detected cards are sent to a device that vibrates to discretely let the wearer know which cards are next from the top of the stack

2 Architecture

In this section, the technical side of the project is explained.

2.1 Finding the codes

First, the Raspberry Pi locates the stack of cards and cuts out the section that contains the code. This is done by finding points with high contrast and removing outliers. Then a box is placed around these points using two smaller bounding boxes. This image is then translated to a black and white rectangle that can be used for reading and translating the codes.

2.2 Translating the codes

Then the Raspberry Pi reads the codes and translates them to numbers between 0 and 51. First, it checks each horizontal line of pixels, and tries to find a card corresponding to that line. This is done by using the two black lines at the start to find out the width of the blocks. After doing this the algorithm filters out all None values and combines multiple of the exact same cards that are next to each other into one.

2.3 Communicating

After that the Raspberry Pi sends the numbers to the Photon and app by hosting a JSON web page. This page contains the deck as a string of numbers, and a human readable version of the deck. The web page is only visible from devices on the same network, as it is not necessary to be able to access this page from another location.

2.4 Buzzing

Finally, the Photon translates the received numbers to the buzzing patterns and the buzzing is performed. The Photon receives a string of numbers between 0 and 51 and translates this to two separate numbers; the first number corresponds to a suit (0-3) and the second to a rank (0-12). Then these numbers are processed by a function that decides the buzzing durations and intervals that correspond to the right card. Finally, it outputs the buzzing patterns to the vibration motor connected to the Photon.

2.5 App

Finally, the app translates the received numbers back to cards and shows the user who has what cards and who will come first, second and third.

3 User Manual

This section describes the needed hardware and how to set it up. It also explains how to get the necessary code on the right hardware and how to run it.

3.1 Hardware

The hardware needed for this project is:

- Stack of cards with the encoding used in this project
- LAN cable connected to the internet
- Mouse with a type A USB connector
- Keyboard with a type A USB connector
- Monitor with HDMI socket
- HDMI cable
- 5V micro USB power supply cable
- Raspberry Pi 2
  https://www.raspberrypi.org/documentation/
- Raspberry Pi Camera Module v2
  https://www.raspberrypi.org/documentation/hardware/camera/
- Raspberry Pi Camera Mount
  https://shop.pimoroni.com/products/raspberry-pi-camera-mount
- Particle Photon IoT Wi-Fi Board
  https://docs.particle.io/photon/
- Grove Base Shield for Photon
  http://wiki.seeedstudio.com/Grove_Base_Shield_for_Photon/
- Grove Vibration Motor
  http://wiki.seeedstudio.com/Grove-Vibration_Motor/
- 5cm Grove Cable
  https://www.seeedstudio.com/Grove-Universal-4-Pin-Buckled-5cm-Cable-5-PCs-Pack.html

3.2 Assembly

This section describes the steps required to assemble and prepare the hardware for the project.

3.2.1 Raspberry Pi and Camera

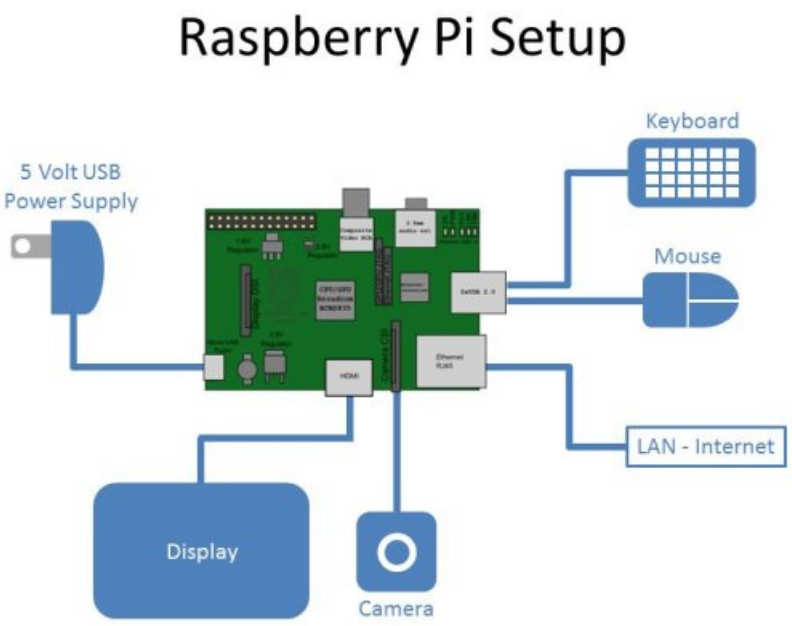

Figure 2: Source: https://projects-raspberry.com/raspberry-pi-motion-sensitive-camera/

1. Connect the USB power supply, keyboard, mouse, LAN internet cable and camera to the Raspberry Pi as shown in Figure 2.
2. Attach the camera mount to the camera as explained at https://shop.pimoroni.com/blogs/help/7987155-assembling-the-camera-mount.
3. Connect the camera to the Raspberry Pi as explained at https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/4.
   Warning: only (dis)connect the camera when the Raspberry Pi is turned off.
4. Update the Raspberry Pi by opening a terminal and typing `sudo apt-get update && sudo apt-get upgrade`.
5. Install the picamera library on the Raspberry Pi by typing `sudo apt-get install python-picamera python3-picamera` in the terminal.
6. Copy all files in the project root folder to the same folder on the Raspberry Pi.
7. To install the required Python dependencies, change the working directory to the folder you copied these files to and type `pip install -r requirements.txt`

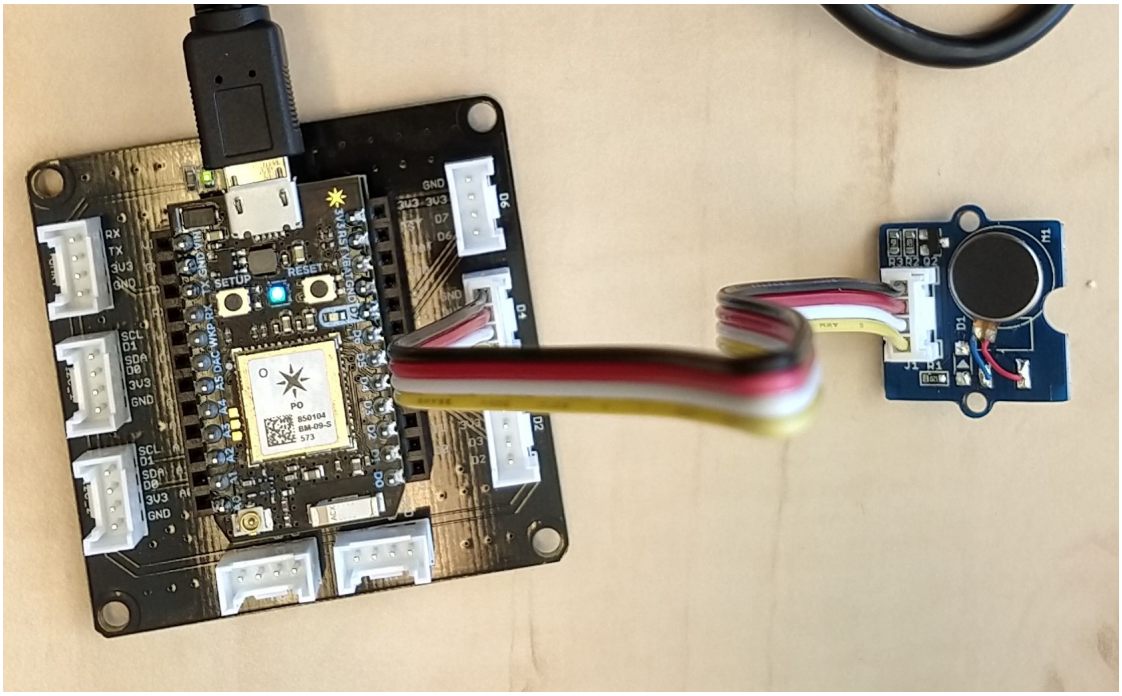

Figure 3: Picture of the assembled Grove and Photon

3.2.2 Photon and Grove

1. Assemble the Photon, USB power supply, Grove cable, Grove vibration motor and Grove Base Shield like in Figure 3. The cable connecting to the vibration motor should go in port D4 of the base shield.
2. Setup the Photon by making an account and following the tutorial on https://setup.particle.io/.
3. Enter https://go.particle.io/shared_apps/5cfa4df733be4a000aba0770 in the browser and select `Copy This App`.
4. Edit the variable `noCards` to change the limit on how many cards the Photon will transmit by buzzing.
5. Save the application and flash it to the Photon by clicking on ϟ (`Flash`).

3.2.3 Connecting

1. Go to https://build.particle.io/build and select `Settings`. Copy the Personal Access Token and paste it in the `access_token` field in the file `jsonread.py`.
2. Then select `Devices` and expand the Photon that has been set up. Copy the Device ID and paste it in the `photon_id` in the file `jsonread.py`.

3.2.4 Running the code

In a terminal on the Raspberry Pi, run `runall.sh` to start the camera and process the incoming visuals to a string of numbers, post the produced string on a website, and send the processed

information to the Photon. In `livefeed.py`, if `print_info` is set to True, information about the deck will be printed to the terminal, and if `show_image` is set to True the image and cropped deck feeds will be shown.

3.3 App

1. Launch the app.
2. Enter the IP address of the Json web page, this address can be found in the output of the raspberry pi.
3. Deal the cards in the order they will be dealt in the game.
4. Press *Done*.

4 Video Demonstration

In the video you can see first the Raspberry Pi doing its work. Our programs are finding the deck of cards and then trying to distinguish each individual card from the deck. Next you can see the working of the JSON page. It displays the cards in the order which the raspberry pi sees them from the top of the stack to the bottom. In the following shot you can see the Photon giving feedback to the user. This is done by activating a buzzer which buzzes according to the code each card is given. There is a pause between the coding of the suit and the rank, and a slightly larger pause between each individual card. As soon as the raspberry pi has processed the image of the deck of cards, the app can be used. This is shown in the next shot. Inside the app, cards can be dealt in the same way as in an actual real life game of poker. When all cards are dealt, the user can press 'done' which displays all cards and determines which player would have won that round.

5 Software

The code of the project can be found on the project Git.