Senior Design Project:

Garage Door Opener based on Image Processing

Coordinator:

Dr. Ahmad Khayyat

Supervisor:

Dr. Ahmed Al-Mulhem

By:

Mohammed Bashmmakh
200973670

Table of Contents

1 Introduction

These days, almost everything now tends to be moving toward automating. people were used to to deal with everything manually, for example, people used to open the garage manually, which means that users had to stop the car, leave the car, then go to the garage door and open it by their hands. After that, the invention of the remote controlled garage doors has caused a great impact on making the lives of the consumer easier. However, as technology improves the lives of the consumers become easier. Thus, this system is aiming to have the garage door to open automatically without having the hands of the suer leave the steering wheel. The system approaches the same idea in an easy and automated way by recognising the car's plate. One of the biggest advantages of automation is ensuring the quality and consistency of the product without forgetting the important aspect which security. The project is going to automate the functionality of the garage systems by using a unique sign for opening the garage door. In other words, each individual car has its own unique plate which is going throw identification and security processes.

2 Problem Statment

The design and the implementation of a garage door opener. Unlike the traditional garage opener that uses a remote control, the system analyzes images of approaching cars and opens the door when a recognized car plate is identified.

3 Project Specifications

3.1 System requirement:

- Accurate identification of car plates.
- The gate opens in a short time.
- Notify the user when a recognition is succeed or failed.
- An automatic way to exit the garage

3.2 Specifications:

- Stay ideal until a car approaches the garage.
- Take a photo of the plate when the car is close to the garage.
- Processing the photo should take a maximum of 3 seconds.
- If a plate is recognise or not a LED will notify the user with different patent indicating whether the plate is recognised or not.
- There will be sensors on the interior side of the garage in order to open it when exiting.

4 System Design

4.1 Architecture

4.1.1 Sub-function identification

The Project was divided into four sub-functions:

Car Approaching

This function is responsible to notify the system that there is a car approaching the garage door, and to operate the System.

Recognition

This is the heart of the project. As it is responsible to capture and recognizing the plate.

Comparison

This function is about comparing different cars' plates that want to enter the garage with registered plates. In addition, plates which are not saved within the system will be denied and will not be permitted to enter.

Regestration

it this function is responsible of registering a new plate to the system.

4.1.2 System architecture and component

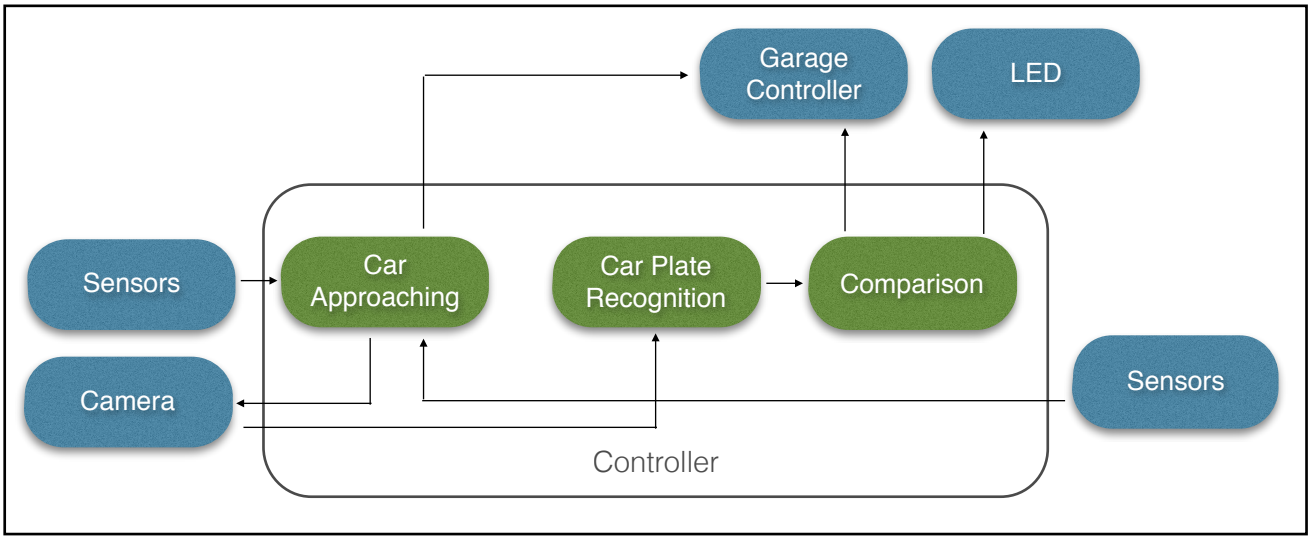

Figure 1: System architecture

4.1.3 Functions of each component

1. Sensor: To sense if a car is at the gate.
2. Camera: To capture an image containing the car's plate.

3. Controller: figure out the best position for the car to capture the image, after that processes it in order to recognize the plate, and finally compare the result with the registered plates, then send a signal to the garage controller.
4. LED: Informing the user whether the plate has been reconized or not, and read the signal which was sent to the garage controller.

4.2 Design Decisions

Controller

The controller has been chosen to be Raspberry Pi, since it is capable of processing the image and identify it, and the advantages are great for the project.

Controller	Advantages	Disadvantages
Raspberry pi	Full OS Support multi developing languages Many types of ports High processing power Fully documented	Few number of pins OS drawbacks Need of system calls
Arduino	Use C Plenty of digital pins Accessories Fully documented	Low processing power Memory limitation Relatively expensive
CMUcam3	Small built in Camera	Poor Documentation

Table 1: Controller Designing Options

Car Plate Recognition

There are two ways of identifying the plate either by reconizing the characters on the plate or by comparing plates' images. The optimal method on comparing plates' images will take too much of time, processing power and memory usage, since the first step is to locate the plate then segment it, meaning that the system will cut image of the plate of the car, and store it with a specific size to compare it later with an approaching cars plate. Also, when comparing, the program has to do the previous steps then compare the images, which requires a lot of processing and power consumption. While characters recognition is easier to implement and will take less processing and power consumption, since there is no need to segment the image and then store it. And for comparing, the system will compare to strings only rather than two images.

Sensing

There multiple method to sense a car is at the garage. Using the camera, by processing the image that has been token every a short period of time, 5 second for example. This method uses a lot of processing and memory usage. Another method, Using approaching sensors or pressure plate as an indicate of car existing at the gate. This solution is better since it is simple and it needs no processing so it saves time and processing power.

4.3 Component Design and Implementation

4.3.1 Off-the-shelf hardware and software components

Raspberry Pi Model B

Raspberry Pi is a credit card-sized single-board computer developed in the UK. Model B is the higher-spec variant of the Raspberry Pi, with 512 MB of RAM, two USB ports, a 100mb Ethernet port and powered by 700 MHz ARM1176JZF-S core.

Figure 2: Raspberry Pi Model B

Components	Specifications
Operating system	Linux (Raspbain)
Power	3.5 W
CPU	700 MHz ARM1176JZF-S
Memory	512 MB
Storage	SD card slot
Graphics	Broadcom VideoCore IV

Table 2: Raspberry Pi Model B specs

Raspberry Pi camera module

a 5MP camera module capable of 1080p video and still images and connects directly to Raspberry Pi. Connected through the CSI (Camera Serial Interface) port on Raspberry Pi. The board is tiny, at around 25mm x 20mm x 9mm and weighing in at just over 3g.

Figure 3: Raspberry Pi Camera module

Pressure Plate

This plate will help to identify the system that there is a car at the gate, by just sending a signal to the Raspberry Pi.

LED

LED will be used as notifier for either if the plate is recognized or not and as reader for the garage controller signal.

Tesseract

Tesseract is an optical character recognition engine for various operating systems It is an open source library. Tesseract is considered one of the most accurate open source OCR engines currently available. To install the library into the Raspberry Pi type “`sudo apt-get install tesseract-ocr`" on the terminal and to use it from the terminal type “`tesseract image.jpg txtFile`”, where image.jpg is the image that will be processed to get the characters form it and txtFile containing the result with a txt format. If you want to use Tesseract directly with in the code, there webpage is fully documented.

ImageMagick

ImageMagick is a free and open source software suit for converting and editing raster image. It is capable of dealing with over 200 image file formats. It is used on reducing the size of the image taken by the Raspberry Pi camera, since the image is large, Tesseract takes more time to processes it. Reducing the size of the image will decrease the time that tesseract usually takes by almost a half. To Install the library and use it on the terminal the user has to type "`sudo apt-get install imagemagick`", then to resize the image type "`convert A.jpg -resize 50% B.jpg`", where A is the targeted image and B the resulting, and the percentage is the resizing ratio of the image, this command will make the image 50% smaller.

4.4 System Integration

4.4.1 Custom software components

Raspberry code

The system has specific tasks, and finding a code that meets all of the requirements is difficult. Using Tesseract OCR library saves a lot of time, since OCR is a big topic.

Raspberry Pi pseudocode

While forever

- if the car is at the gate
  - Capture an image
  - process it with Tesseract OCR
  - Compare the result
    - If the plate is registered
      - Open the garage
      - Turn LED pattern
    - Else don't open
      - Do not open the garage
      - Turn LED pattern
- If Registered
  - Capture an image
  - process it with Tesseract OCR
  - Register new plate

4.4.2 Interfaces between components

The system has two interfaces:

1. Camera to Raspberry PI: dedicated CSI interface, which was designed especially for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data.

Figure 4: Raspberry Pi and Camera module connected

2. Sensors and LEDs to Raspberry PI: through GPIO (general purpose input/ output) pins on the Raspberry PI .

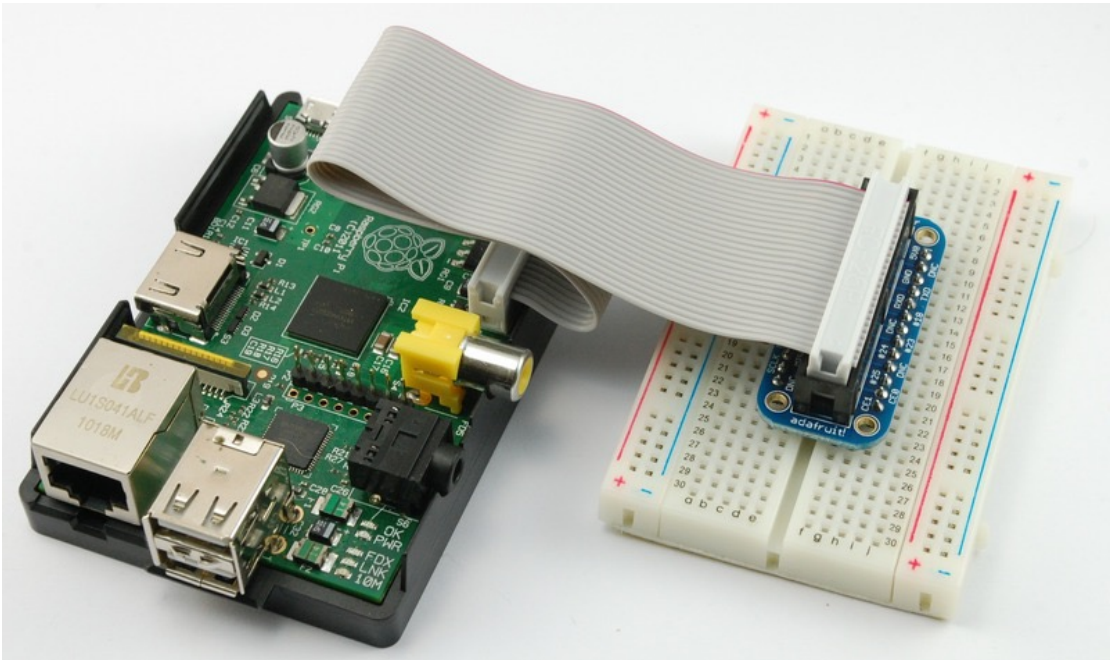

Figure 5: Raspberry Pi GPIO connected to Breadboard

5 Testing and Evaluation

5.1 Testing

In the testing phase, Everything has to work as intended expect Tesseract OCR. The system was tested with a custom plates, a white paper with 3 characters and 3 numbers, because the version of Tesseract that installed on the Raspberry Pi does not recognise arabic characters probably; thus, Tesseract could not recognize saudi car plates. As Tesseract faces the arabic letters first and does not recognised it, it ignores the rest. In other words, when it tries to process Saudi plates it gives a plank text. However when Tesseract processes Dubai's plates it recognised it.

5.2 Evaluation

The system achieved the requirements; however there is a security vulnerability. The prototype does not prevent fake attempts to the garage, as the system is using characters recognition only as an identification method it can be deceived by a fake plate, or a paper that has the characters of the registered plate. To fix this issue, solutions can include using the image matching or studying the depth of the image (Depth Map). That means studying the information that is related to the distance of the surfaces of scene objects from a viewpoint, or use a two step verification, for example, using RFID tag in addition to plate recognition.

6 Engineering Tools and Standards

1. Tesseract library.
2. ImageMagick library
3. Python as programming language

7 Issues

7.1 Problems and Challenges

All the project problems are listed in the following tables, including the unsuccessful attempts and final solution:

#	The issue	Unsuccessful attempted	Cause of failure	Final resolution
1	Using CMUcam3 as a processing unit	implementing and running a program	Poor documentation and bad decisions on choosing the parts based on availability	switch to raspberry pi as a processing after wasting a week and half
2	Using motion sensor	it senses that there is an object always	Long rage	use a pressure plate

3	Ultra sonic sensor	conect it to raspberry pi	Raspberry pi does not have an ADC.	use a pressure plate
4	Recognize Arabic letters	Tesseract does not recognise Suadi's Plate	Tesseract fails on recognition Arabic letters	Making a custom plates

7.2 Limitations and constraints of the design

The project is a standalone system with no interface to communicate with other systems, so any additional communication with this system need an extra effort to make it work. For example, if the implantation wants another sensing rather than a sensor that send either high or low voltage as an output there will be some modification on the program.

7.3 Limitations and constraints of the implementation

There was constrains on the implementation which is using open sources libraries. The only limitation on implementation is the security vulnerability that has been mentioned above and it can be solved.

8 Conclusion

To sum up, this project is going to attempt make the consumer's life easier by improving the traditional remote controlled garage. This report has identified requirements, specification, advantages and disadvantages of the product.In addition, limitations of developing this product has been encountered, and will be, hopefully, solved in the future. Furthermore, I have learned how to develop applications on Python. Getting to know how to use and develop using raspberry pi was such a great experience to have. Using third-party libraries to perform specific tasks and enhance the output. It was a bad decision to work alone, being alone have bad impacts, lake of creativity, sharing ideas.

APPENDIX A

Raspberry Pi code

```
#imported library
import RPi.GPIO as GPIO
import os
import time
import picamera

GPIO.cleanup()
GPIO.setmode(GPIO.BCM)

# pins setup
LED=23
GPIO.setup(LED, GPIO.OUT)
#setup pin 22 as pull down to check later if it is pressed the
plate will return High
PrussurePlate=22
GPIO.setup(PrussurePlate, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

#Stop the process
Stop=18
GPIO.setup(Stop, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

Register = 24
GPIO.setup(Register, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

camera = picamera.PiCamera()
camera.hflip = True
camera.vflip = True

RegisteredPlate = "ABC123"
############

########
t=1
while t:
      #if PrussurePlate is pressed
      if GPIO.input(PrussurePlate)== True:
            # Capture an image
            camera = picamera.PiCamera()
            camera.hflip = True
            camera.vflip = True
            print( "capturing an image" )
            camera.capture('image1.jpg')

            # Reducing image size to half
            print ("resizing the image")
```

```
            os.system(" convert image1.jpg -resize 50%
image2.jpg ")

            # runnung tesseract
            print( "Running Tesseract " )
            os.system ("tesseract image2.jpg testing ")
            print ("Tesseract Finished")
            print ("the output is ")
            f = open ("testing.txt", "r")
            result = f.readline().replace(" ","")
            result = result.replace("\n","")
            print (result )

            if result == RegisteredPlate :
                  print ("The Plate has been recognized")
                  GPIO.output(LED, 1)
                  time.sleep(2)
                  GPIO.output(LED, 0)
                  time.sleep(2)

            else :
                  print ("The Plate has not been recognized")
                  GPIO.output(LED, 1)
                  time.sleep(2)
                  GPIO.output(LED, 0)
                  time.sleep(2)
                  GPIO.output(LED, 1)
                  time.sleep(2)
                  GPIO.output(LED, 0)
                  time.sleep(2)

      if GPIO.input(RegisteredPlate)== True:
            # Capture an image
            camera = picamera.PiCamera()
            camera.hflip = True
            camera.vflip = True
            print( "capturing an image" )
            camera.capture('image1.jpg')

            # Reducing image size to half
            print ("resizing the image")
            os.system(" convert image1.jpg -resize 50%
image2.jpg ")

            # runnung tesseract
            print( "Running Tesseract " )
            os.system ("tesseract image2.jpg testing ")
            print ("Tesseract Finished")
```

```
            print ("the output is ")
            f = open ("testing.txt", "r")
            result = f.readline().replace(" ","")
            result = result.replace("\n","")
            print (result )
            # if the plate recognized
            if result != "" :
                  print ("Plate information was : " + Register +
" and has been updated to : " + result )
                  Register = result
            else :
                  print ("Plate did not recognized")

      if GPIO.input(Stop)== True:
            print ("the Program stoped, Goodbye")
            t=0
            GPIO.output(LED, 1)
            time.sleep(2)
            GPIO.output(LED, 0)
            time.sleep(2)

GPIO.cleanup()
```

APPENDIX B

Long Term Plan

Task Mode	Task Name	Duration	Start	Finish
	Collecting the Requirments	**20 days**	**Tue 16/09/14**	**Mon 13/10/14**
	Collecting the Requirments	7 days	Tue 16/09/14	Wed 24/09/14
	Review with the adviser	7 days	Thu 25/09/14	Fri 03/10/14
	Researching the idea	4 days	Sat 04/10/14	Wed 08/10/14
	Finalize the Requirments	3 days	Thu 09/10/14	Mon 13/10/14
	Designing	**20 days**	**Tue 14/10/14**	**Sun 09/11/14**
	Architecture Design	6 days	Tue 14/10/14	Tue 21/10/14
	Descussing the Design	7 days	Wed 22/10/14	Thu 30/10/14
	Finalize the hardware compnents	7 days	Fri 31/10/14	Sun 09/11/14
	Implementation	**17 days**	**Mon 10/11/14**	**Tue 02/12/14**
	Collect the devices	5 days	Mon 10/11/14	Fri 14/11/14
	Coding	7 days	Sat 15/11/14	Mon 24/11/14
	Applying the codes on the devices	6 days	Tue 25/11/14	Tue 02/12/14
	Testing and debuging	**16 days**	**Wed 03/12/14**	**Wed 24/12/14**
	Debuging the microcontrollers	7 days	Wed 03/12/14	Thu 11/12/14
	Code tracing	5 days	Fri 12/12/14	Thu 18/12/14
	troubleshooting	4 days	Fri 19/12/14	Wed 24/12/14

APPENDIX C

commands to deal with GPIO in Python

```
# import RPi.GPIO module
import RPi.GPIO as GPIO
# choose BOARD or BCM
# BCM for GPIO numbering
GPIO.setmode(GPIO.BCM)
# BOARD for P1 pin numbering
GPIO.setmode(GPIO.BOARD)
# Set up Inputs
# set port/pin as an input
GPIO.setup(port_or_pin, GPIO.IN)
# input with pull-down
GPIO.setup(port_or_pin, GPIO.IN,  pull_up_down=GPIO.PUD_DOWN)
# input with pull-up
GPIO.setup(port_or_pin, GPIO.IN,  pull_up_down=GPIO.PUD_UP)
# Set up Outputs
# set port/pin as an output
GPIO.setup(port_or_pin, GPIO.OUT)
# set initial value option (1 or 0)
GPIO.setup(port_or_pin, GPIO.OUT, initial=1)
# Switch Outputs
# set an output port/pin value to 1/GPIO.HIGH/True
GPIO.output(port_or_pin, 1)
# set an output port/pin value to 0/GPIO.LOW/False
GPIO.output(port_or_pin, 0)
# Read status of inputs OR outputs
# read status of pin/port and assign to variable i
i = GPIO.input(port_or_pin)
# use input status directly in program logic
if GPIO.input(port_or_pin):
# Clean up on exit
GPIO.cleanup()
# What Raspberry Pi revision are we running?
#  0 = Compute Module, 1 = Rev 1, 2 = Rev 2, 3 = Model B+
GPIO.RPI_REVISION
# What version of RPi.GPIO are we running?
GPIO.VERSION
# What Python version are we running?
import sys; sys.version
```