Face recognition treasure box using Raspberry Pi

Khyathi Balusu

BVRIT Hyderabad

18th April,2015

Introduction

- treasure box which unlocks itself using face recognition running on a Raspberry Pi.
- Face recognition to be done accurately, by differentiating between photos and actual face

Raspberry Pi

Processor	Broadcom BCM2835 SoC (System on Chip)
Core	ARM11 (700 MHz ARM1176JZF-S)
Memory (RAM)	256 Mb (Model A) or 512 Mb (Model B/B+)
GPU	Broadcom Videocore IV
USB 2.0	1 (Model A) / 2 (Model B) / 4 (Model B+)
On Board Storage	SD Card (Model A/B) / Micro SD (Model B+)
Video Input	Camera Support via CSI Connector
Video Output	Composite Video (RCA) or HDMI or LCD Support
Networking	10/100 Mbit/s Ethernet Support on Model B/B+
Audio Outputs	3.5 mm Jack
I.O. Lines	**26 Pin GPIO** Connector on Model A/B and **40 Pin GPIO** Connector on Model B+
Power Source	5V/2A DC Adapter via **Micro USB**
Power Ratings	300 mA (1.5W – Model A), 700 mA (3.5W – Model B) & 600 mA (3W – Model B+)

Hardware and Software

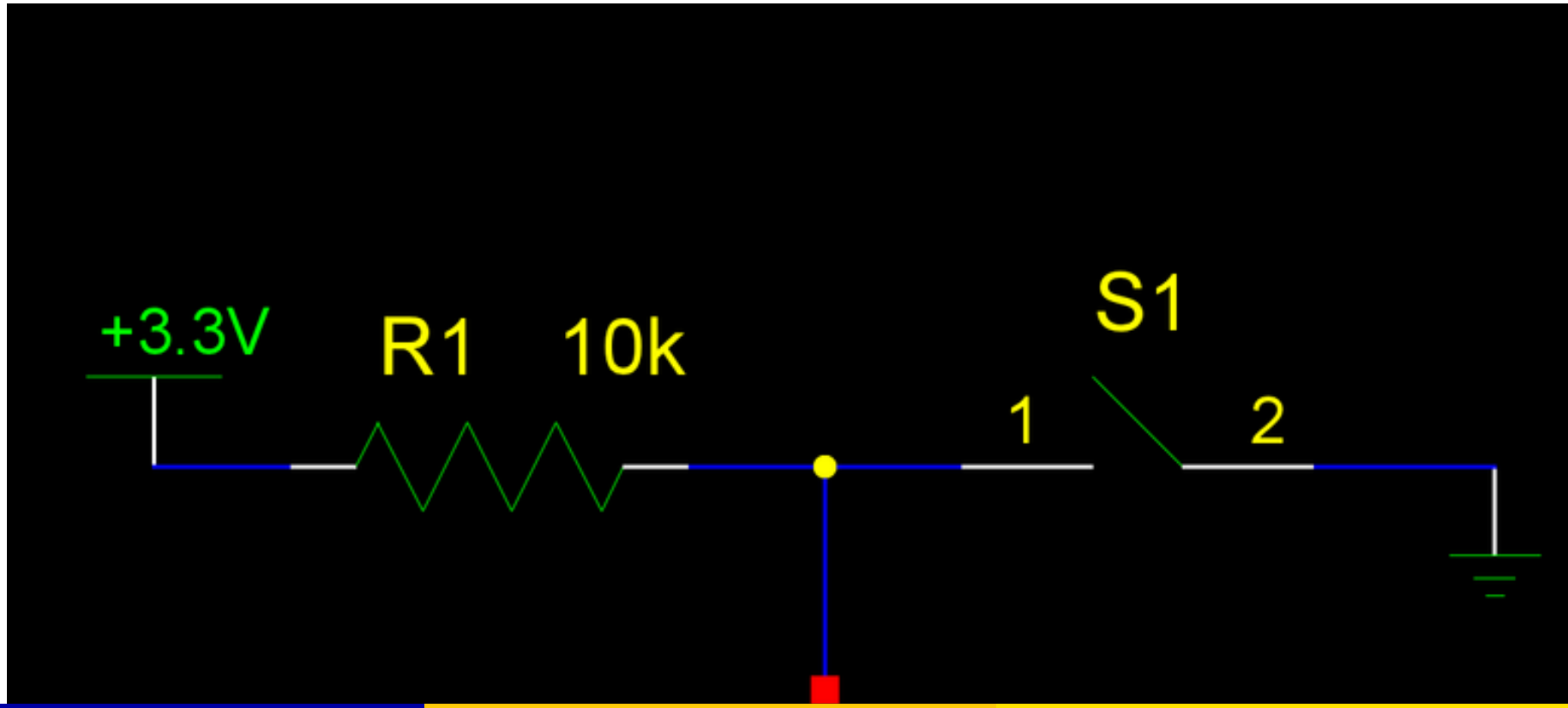
- Hardware required :Raspberry Pi, RPi camera, box, servo with lock solenoid, power supply, hookup wires
- Software to be used : OpenCV, Python, Picamera library

Hardware

- Raspberry Pi, running the Raspbian operating system.
- Raspberry Pi camera
- Small box that can fit the Raspberry Pi and locking mechanism inside.
- Small servo or lock solenoid for the locking mechanism.
- Momentary push button that can mount to the box.
- Power supply for the Raspberry Pi and servo or solenoid. For powering a micro servo, a 4x AA battery pack is a simple option.
- Hookup wires to connect the switch, servo, and servo power supply.

Wiring

- For the servo, connect the signal line to GPIO 18 of the Raspberry Pi.
- The push button is attached to GPIO 25 of the Raspberry P
- Each button connection looks like: 3.3v – 10k Pull-up Resistor – GPIO – Button –GND



Software

- This project depends on the OpenCV computer vision library to perform the face detection and recognition.

Commands for OpenCV:

- sudo apt-get update
- sudo apt-get install build-essential cmake pkg-config python-dev libgtk2.0-dev libgtk2.0 zlib1g-dev libpng-dev libjpeg-dev libtiff-dev libjasper-dev libavcodec-dev swig unzip
- wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.9/opencv-2.4.9.zip
- unzip opencv-2.4.9.zip

- cd opencv-2.4.9
- make
- sudo make install
- cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local -DBUILD_PERF_TESTS=OFF -DBUILD_opencv_gpu=OFF -DBUILD_opencv_ocl=OFF

Formatting and booting

- For formatting: Minimum 4GB SD card. Software : SDFormatter
- For booting: Software- Win32DiskImager
- Time taken : Approximately 15 minutes.

Python Dependencies

- sudo apt-get install python-pip
- sudo apt-get install python-dev
- sudo pip install picamera
- sudo pip install rpio

Work Update

- Setting up the Pi, booting the OS, setting up libraries successfully completed
- Major components purchased and tested. Yet to place inside the box permanently.
- Coding for camera capture, lock and hardware done.
- OpenCV code for training downloaded.Yet to start training

Challenges till date

- Display for the Pi, HDMI compatibility
- LAN port not being recognized
- Trouble purchasing correct servo

Code for image capture

switchcamera.py - SciTE

File Edit Search View Tools Options Language Buffers Help

1 switchcamera.py | 2 threeswitchlcd.py | 3 spibcm.py | 4 serial1.py | 5 1.pgm

```
import time
import picamera

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(7,GPIO.IN)
with picamera.PiCamera() as camera:
    camera.start_preview()
    while True:
        if(GPIO.input(7)==0):
            camera.capture('/home/pi/Surya/image.jpg')
            camera.stop_preview()
```


8:02 PM
4/17/2015

Code for servo

servo.py * SciTE

File Edit Search View Tools Options Language Buffers Help

1 servo.py *

```
Raspberry Pi Face Recognition Box Servo Calibration Sketch


from RPIO import PWM

import config

servo = PWM.Servo()

print 'Servo Calibration'
print
print 'Use this tool to find the pulsewidth values which move the'
print 'lock latch to the locked and unlocked position. Update config.py'
print 'with the locked and unlocked servo pulsewidth values.'
print
print 'Values range from 1000 to 2000 (in microseconds), with 1500 being the center.'
print
print 'Press Ctrl-C to quit'
print

while True:
    val = raw_input('Enter servo pulsewidth (1000 to 2000):')
    try:
        val = int(val)
    except ValueError:
        print 'Invalid value, must be between 1000 and 2000!'
        continue
    if val < 1000 or val > 2000:
        print 'Invalid value, must be between 1000 and 2000!'
        continue
    servo.set_servo(config.LOCK_SERVO_PIN, val)
```


8:03 PM
4/17/2015

Code for hardware

hardware.py - SciTE

File Edit Search View Tools Options Language Buffers Help

1 hardware.py

```
import RPIO
from RPIO import PWM

import picam
import config
import face


class Box(object):
    """Class to represent the state and encapsulate access to the hardware of
    the treasure box."""
    def __init__(self):
        # Initialize lock servo and button.
        self.servo = PWM.Servo()
        RPIO.setup(config.BUTTON_PIN, RPIO.IN)
        # Set initial box state.
        self.button_state = RPIO.input(config.BUTTON_PIN)
        self.is_locked = None

    def lock(self):
        """Lock the box."""
        self.servo.set_servo(config.LOCK_SERVO_PIN, config.LOCK_SERVO_LOCKED)
        self.is_locked = True

    def unlock(self):
        """Unlock the box."""
        self.servo.set_servo(config.LOCK_SERVO_PIN, config.LOCK_SERVO_UNLOCKED)
        self.is_locked = False

    def is_button_up(self):
        """Return True when the box button has transitioned from down to up (i.e.
        the button was pressed)."""
        old_state = self.button_state
        self.button_state = RPIO.input(config.BUTTON_PIN)
        # Check if transition from down to up
        if old_state == config.BUTTON_DOWN and self.button_state == config.BUTTON_UP:
            # Wait 20 milliseconds and measure again to debounce switch.
            time.sleep(20.0/1000.0)
            self.button_state = RPIO.input(config.BUTTON_PIN)
            if self.button_state == config.BUTTON_UP:
```


8:03 PM
4/17/2015

Code for training

train.py * SciTE

File Edit Search View Tools Options Language Buffers Help

1 train.py *

```
import fnmatch
import os

import cv2
import numpy as np

import config
import face


MEAN_FILE = 'mean.png'
POSITIVE_EIGENFACE_FILE = 'positive_eigenface.png'
NEGATIVE_EIGENFACE_FILE = 'negative_eigenface.png'


def walk_files(directory, match='*'):
    """Generator function to iterate through all files in a directory recursively
    which match the given filename match parameter.
    """
    for root, dirs, files in os.walk(directory):
        for filename in fnmatch.filter(files, match):
            yield os.path.join(root, filename)

def prepare_image(filename):
    """Read an image as grayscale and resize it to the appropriate size for
    training the face recognition model.
    """
    return face.resize(cv2.imread(filename, cv2.IMREAD_GRAYSCALE))

def normalize(X, low, high, dtype=None):
    """Normalizes a given array in X to a value between low and high.
    Adapted from python OpenCV face recognition example at:
      https://github.com/Itseez/opencv/blob/2.4/samples/python2/facerec_demo.py
    """
    X = np.asarray(X)
    minX, maxX = np.min(X), np.max(X)
    # normalize to [0...1].
    X = X - float(minX)
    X = X / float((maxX - minX))
    # scale to [low...high].
```


8:04 PM
4/17/2015