SECURE QUADCOPTER DELIVERY

***Project Reference No.:** 45S_BE_3528*

***College** : New Horizon College of Engineering, Bengaluru*
***Branch** : Department of Computer Science and Engineering*
***Guide(s)** : Ms. Soja Rani S*
Dr. Vaishali M Deshmukh
***Student(S)** : Ms. Shreya Pradeep*
Mr. Rahul Musaliyath Dines
Mr. Shivanand G Prabhu

Keywords: Autonomous delivery vehicles, Unmanned Aerial Vehicles, Drones, Delivery robot, secure authentication system, Voice recognition, Android App Development, REST APIs, Raspberry Pi.

Introduction:

Autonomous delivery services have been at the forefront of retail and e-commerce industries in the recent past. Higher efficiency, the ability to deliver in remote places, and sustainability are some of the many features of autonomous delivery systems that have boosted their growth in various fields. Drones or quadcopter are also being used as agents of autonomous delivery systems. Drone technology is a rapidly growing innovation which has witnessed exponential growth in various industries including e-commerce, healthcare, hospitality and in disaster management and recovery. Features including low cost, easy manoeuvring, increased customer satisfaction and minimal operational requisites have helped commercial establishments, government institutions and hobbyists to realise the tremendous potential of drones. The Phase-2 of this project aims to build a model that enables autonomous delivery to be more secure. The model is designed to implement an authentication mechanism by employing a One-Time Password, that verifies the delivery recipient. This system enhances the delivery service by eliminating the risk of incorrect deliveries and other security threats. The addition of this security layer to autonomous delivery systems can make them less dependent on human verification and enhance the security aspects of the delivery.

Background:

In the Phase-1 of our project, we published an IEEE review paper titled, "Securing Technology Enabled Services Using Unmanned Aerial Vehicles". This paper probes some of the contemporary applications of drones such as in assisting society and succouring victims during natural and man-made calamities, secure package delivery, among others. Several existing solutions which include facial recognition, object detection, visual aided navigation, voice-based control and global positioning system-based tracking, which can enable several sectors to reap the benefits of drone technology are evaluated. This paper also proposes a secure drone delivery model, which we later aim to implement in the Phase-2 of our project, which utilises a voice-based OTP authentication system to securely transport packages.

Objectives:

- To build an authentication system for autonomous delivery systems like drones and robots.
- To employ deep learning modules such as speech recognition.
- To utilize the widely used system of One-Time Passwords to securely authenticate the receiver.
- To provide a simple, easy to use interface for dispatchers to send their package to the receiver.
- To build a cost-effective system to securely authenticate the receiver of the payload.
- To research the existing methodologies used for autonomous deliveries.
- To probe the growing research opportunities in the field of autonomous deliveries and quadcopters.
- To publish the findings and results in a Scopus-indexed journal/in the form an IEEE paper.

Methodology:

Using a remote device or drone, we can deliver packages in an environment friendly manner which provides a greener alternative to traditional last mile delivery. The GPS coordinates of the recipient must be inputted into the autonomous vehicle or drone. The autonomous device then navigates to the destination. It then proceeds for authentication using the voice-enabled security mechanism that is described further.

The package's recipient then recites the one-time password that was supplied to their phone via SMS, app notification or e-mail. The device will show that the authentication has been successful provided that the OTP deciphered from the recipient's audio is identical to the OTP set by the dispatcher. If the authentication is successful, the device then proceeds to further action such as releasing the payload. If the authentication fails, the authentication is considered unsuccessful and a suitable action can then be taken by the autonomous device.

Implementation: We first complete the hardware connections as shown in as shown in Fig. 1. The cathodes of the LEDs are connected to the GPIO pins numbered 18 and 24. The anode of each LED is connected to a 330Ω resistor. The resistors are then connected to any one of the eight ground pins on the Raspberry Pi. The schematic diagram of these hardware connections is shown in Fig. 2. The ROS is installed into a formatted microSD card. This microSD card is then inserted into the Raspberry Pi's microSD slot. The microcomputer is then powered on by connecting it's USB-C power in port to a laptop. The power from the laptop should not be more than 5V 2A. Once the Raspberry Pi boots, we do not require additional Python installation as Python comes preinstalled with ROS. The Thonny IDE is launched and the Python program described previously is coded. The USB mini microphone is then connected to the Raspberry Pi's USB 2.0 slot. This will be used to obtain an efficient voice-based input for speech recognition. The Python program is now run. The microphone then captures the receiver's OTP recital, and authenticates it using the Python code. This security module can be easily integrated with various types of autonomous delivery vehicles like drones, droids or robots by multiple interfaces including

HDMI, Bluetooth, Wi-Fi or through USB ports. When integrated with an autonomous delivery vehicle, further suitable action(s) may be taken following the authentication like payload release. An Android app to be used by the sender to generate and send an OTP through SMS to the receiver, was developed. This OTP generated using the app is then securely hosted online by employing a cloud-based REST API service. After generating the OTP, the app sends an HTTP POST request to the REST API server over an HTTPS connection, with the body containing the OTP as shown in Fig. 3. The Python code on the Raspberry Pi then accesses this newly generated OTP over an HTTP GET request so that it can securely authenticate the receiver. The SMS containing the OTP is sent to the receiver as shown in Fig. 4. The user interface (UI) of the Android app is shown in Fig. 5. Fig. 6 shows the Python code in the Thonny IDE that was developed to securely authenticate the receiver. Fig. 7 shows the actual implementation of the proposed authentication system. Fig. 8 shows the blue LED lighting up for an unsuccessful authentication and the green LED lighting up for a successful authentication.

Results and Conclusions: Autonomous delivery vehicles are becoming increasingly popular and would be the preferred mode for last mile deliveries in the near future. Therefore, the need to enhance these systems with a security mechanism is imperative. The use of a Raspberry Pi aides in various benefits including lowering of operating costs, permitting multiple connection interfaces and providing larger processing power. A Python code can automate operations as shown by the OTP authentication process, thus, providing a simple yet cost effective way to implement a security system for autonomous delivery vehicles.

Scope for future work: This system can later be integrated with a flight controller in a drone or the controller of an autonomous delivery robot to fully deploy the system. The authentication mechanism can later be extended to authenticate the package receivers using facial data or biometrics. The OTP can later be generated and verified by using newer technologies like blockchain.

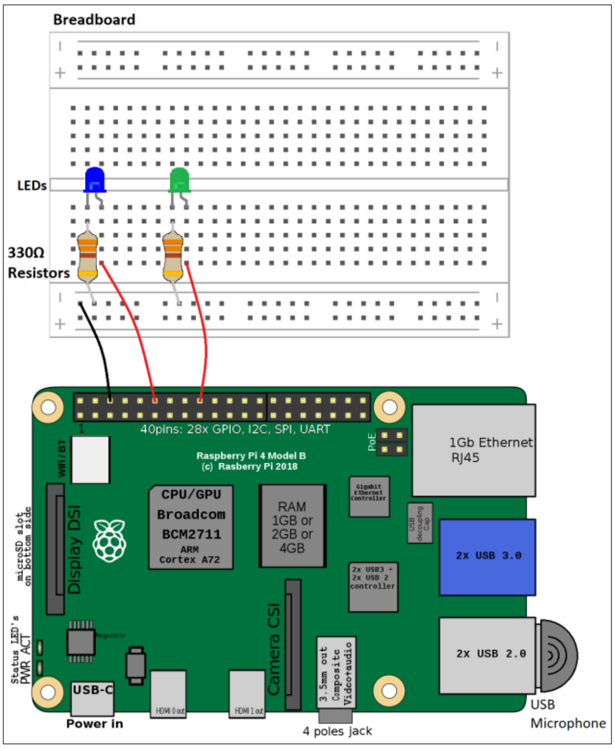
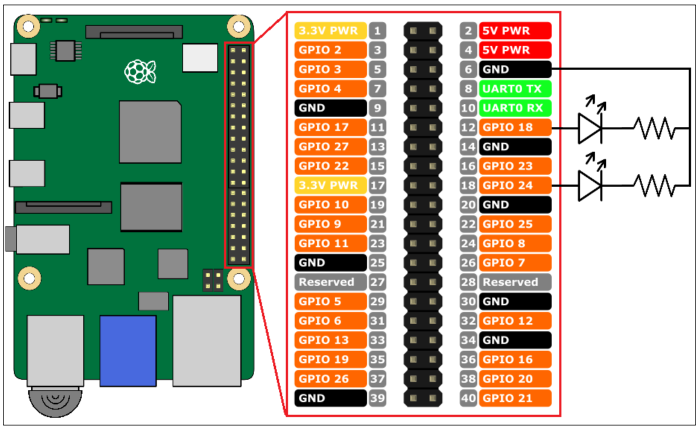


Fig. 1: Pictorial representation of the system



3.3V PWR	1	2	5V PWR
GPIO 2	3	4	5V PWR
GPIO 3	5	6	GND
GPIO 4	7	8	UART0 TX
GND	9	10	UART0 RX
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
3.3V PWR	17	18	GPIO 24
GPIO 10	19	20	GND
GPIO 9	21	22	GPIO 25
GPIO 11	23	24	GPIO 8
GND	25	26	GPIO 7
Reserved	27	28	Reserved
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12
GPIO 13	33	34	GND
GPIO 19	35	36	GPIO 16
GPIO 26	37	38	GPIO 20
GND	39	40	GPIO 21

Fig. 2: Schematic diagram of hardware connections

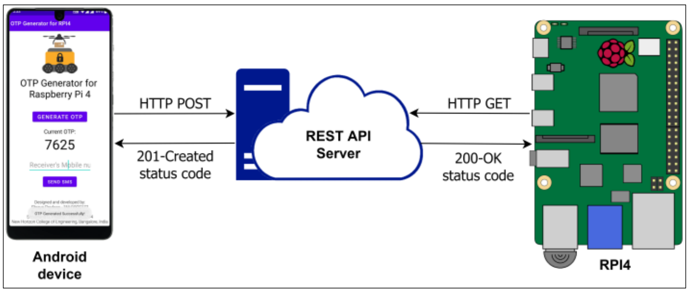
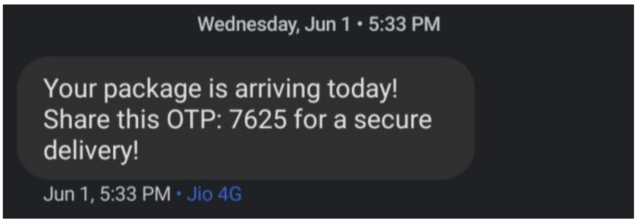


Fig. 3: Using REST APIs for setting the OTP

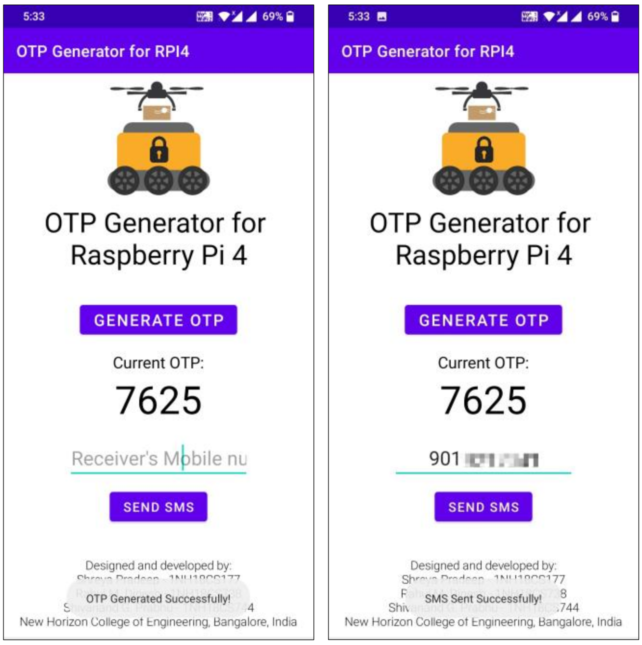


Wednesday, Jun 1 • 5:33 PM

Your package is arriving today! Share this OTP: 7625 for a secure delivery!

Jun 1, 5:33 PM • Jio 4G

Fig. 4: SMS sent to the receiver



OTP Generator for Raspberry Pi 4

Current OTP:

7625

Receiver's Mobile nu

OTP Generated Successfully!

Designed and developed by:
Shreya Pradeep - 1NH18CS177
New Horizon College of Engineering, Bangalore, India

OTP Generator for Raspberry Pi 4

Current OTP:

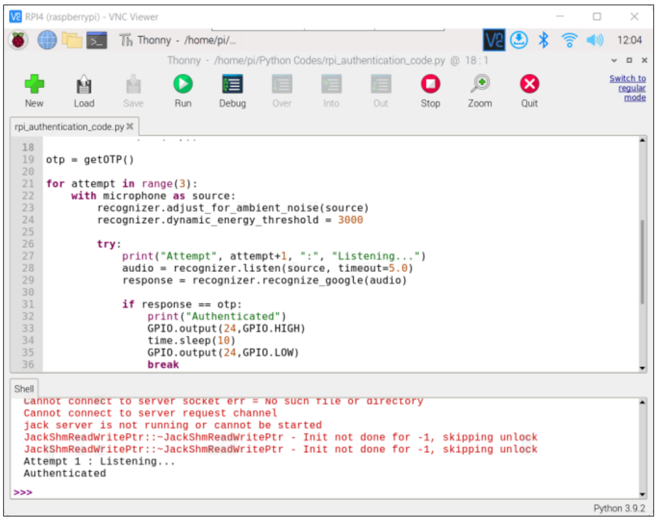
7625

901 [redacted]

SMS Sent Successfully!

Designed and developed by:
Shreya Pradeep - 1NH18CS177
New Horizon College of Engineering, Bangalore, India

Fig. 5: The UI of the Android app developed for generating the OTP that would be used by the authentication system on the Raspberry Pi. An SMS can also be sent to the receiver using the app.


RPI4 (raspberrypi) - VNC Viewer
Thonny - /home/pi/...
12:04
Thonny - /home/pi/Python Codes/rpi_authentication_code.py @ 18 : 1
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Switch to regular mode
rpi_authentication_code.py
Shell
Python 3.9.2

```
otp = getOTP()

for attempt in range(3):
    with microphone as source:
        recognizer.adjust_for_ambient_noise(source)
        recognizer.dynamic_energy_threshold = 3000

        try:
            print("Attempt", attempt+1, ":", "Listening...")
            audio = recognizer.listen(source, timeout=5.0)
            response = recognizer.recognize_google(audio)

            if response == otp:
                print("Authenticated")
                GPIO.output(24,GPIO.HIGH)
                time.sleep(10)
                GPIO.output(24,GPIO.LOW)
                break
```

```
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Attempt 1 : Listening...
Authenticated

>>>
```

Fig. 6: The Thonny Python IDE in ROS executing the Python code used for the authentication system. The console output shows the status of the authentication and is indicated using LED lights.

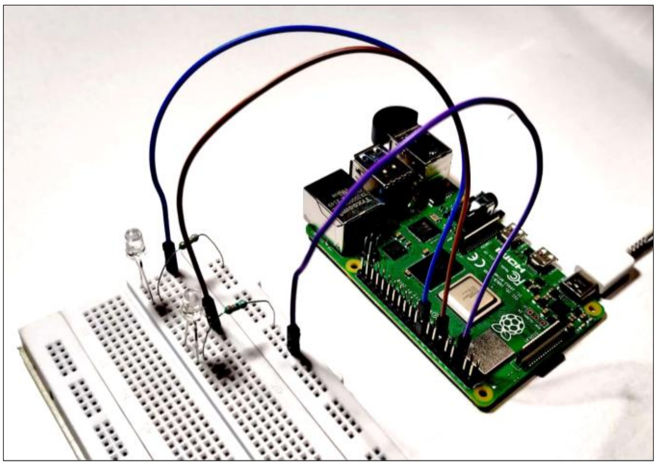


Fig. 7: The Raspberry Pi with USB mini microphone, connected to the breadboard containing resistors and LEDs using jumper wires.

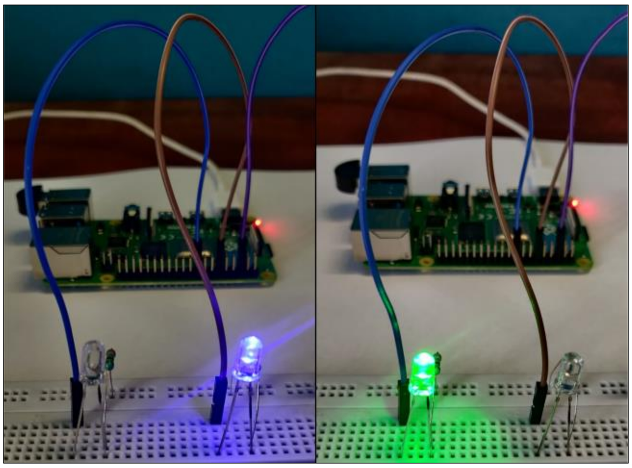


Fig. 8: The blue and green LEDs illuminating to indicate an unsuccessful and successful authentication respectively.