

Article

A Solar-Powered IoT Connected Physical Mailbox Interfaced with Smart Devices

Tareq Khan

School of Engineering, Eastern Michigan University, Ypsilanti, MI 48197, USA; tareq.khan@emich.edu

Received: 10 July 2020; Accepted: 8 September 2020; Published: 15 September 2020



Abstract: In this age of electronic communication, checking the physical mailbox is still part of our daily life. Mailboxes are placed several meters away from the house, and sometimes across the opposite side of a street. It is annoying to walk to the mailbox each day and realize that the mailbox is empty. In this paper, an Internet of Things (IoT)-connected smart mailbox is developed that automatically sends a notification to the smartphone whenever new mail arrives. This notification removes the frustration of an unnecessary trip to the mailbox. The proposed mailbox contains a low power device comprising of sensors, a system-on-chip microcontroller with Bluetooth Low Energy (BLE), and a rechargeable battery that charges with a solar panel. Whenever the mailbox door is closed, it checks the presence of mail and then sends the data to a hub using BLE. The hub is placed inside the house and sends a push notification to the smartphone using the home Wi-Fi. A smart speaker, Google Home, is also interfaced with the system, which can verbally say the status of the mailbox when asked a customized question. A prototype of the smart mailbox, the hub, and the smartphone app is developed and tested successfully.

Keywords: Bluetooth Low Energy (BLE); cloud messaging; Hypertext Transfer Protocol (HTTP) server; infrared reflectance sensor; Raspberry Pi; smart speaker; solar panel; Wi-Fi

1. Introduction

Internet of Things (IoT) and smart home technologies are bringing lots of excitement in our day to day lives. A smart home refers to a convenient home setup where appliances and devices—such as thermostats, lights, window, door, etc.—can be automatically monitored and controlled remotely using a smartphone or other networked device [1–5]. Smart home technology provides homeowners with ease and better accessibility. The global home automation market was valued at about \$24 billion in 2016. It is expected to grow to about \$53.5 billion by 2022 as more people begin to adopt smart home technology [6].

In this age of email and social media, we still look forward to checking our physical mailbox each day. According to a survey by Gallup, seniors are more likely than younger adults to enjoy checking the mail, but 36% of Americans under 30 also feel this way [7]. A question of burning importance is: When to check the mailbox? Sometimes the mailboxes are placed several meters away from the house door and sometimes they are placed on the other side of a busy street. It might be frustrating for people, especially seniors and the disabled, to walk several meters to the mailbox each day by crossing a busy street and find that no mail has arrived. The situation becomes more troublesome when the weather is extremely hot, cold, or rainy. In this project, an IoT connected smart mailbox is developed. It automatically detects when new mail is put inside the mailbox and it sends a notification to the owner's smartphone. The notification informs the owner exactly when the mail has arrived and removes the frustration of an unnecessary trip to the mailbox. The owner is notified of an anticipated letter or gift as soon as it is delivered to the mailbox—as long as the owner's smartphone is connected with the Internet on any place of the world—and it gives peace of mind.

In the proposed method, each smart mailbox contains a low power electronic device comprising of a sensor for detecting the status of the mailbox door, a sensor to detect the presence of items inside the mailbox, a system-on-chip microcontroller with Bluetooth Low Energy (BLE), and a rechargeable lithium-ion polymer battery. The key design challenge of the electronic device is to make it consume ultra-low power so that the battery can be charged using a solar panel without the need for manual recharging. The proposed device is designed with low power consumption in mind and its battery is charged with a solar panel, mounted on top of the mailbox. The device wakes up from low power sleep mode whenever the mailbox door is closed, then it checks the presence of any item in the mailbox and sends the data to a hub unit using BLE. Wireless systems, such as BLE, are easier to install than wired systems. The hub is placed inside the home and it is wirelessly connected with the home Wi-Fi. The hub then sends a push notification using the Internet to the owner’s smartphone. A smart speaker, Google Home, is also interfaced with the system. When the owner asks Google Home: “Check physical mailbox”, it answers “you have mail in your mailbox” or “your mailbox is empty”, depending upon the status of the mailbox. Additionally, the status of the mailbox, whether empty or having mail, can be read using a web browser from any place of the world.

Several related works are found in the literature on developing a smarter mailbox. A comparison with related works is shown in Table 1. In [8], a mail alert system is developed for centralized mailbox systems using a programmable logic controller (PLC) and Global System for Mobile (GSM) modem. Email or short message service (SMS) is generated to notify the user. However, the current consumption of GSM modem in normal operation is 250 mA and can rise to 1 A while transmission [9]—thus, it will require wiring of external power supply to the mailbox as the battery will not last long. Moreover, the GSM modem will require a subscriber identity module (SIM) card, and the extra monthly payment will also be required. The work in [10] is focused on digitizing building addresses and storing them in Radio-frequency identification (RFID) tag, which can be used for drone delivery. The proposed mailbox uses a passive infrared (PIR) motion sensor, cameras, solenoid locks, and RFID reader controlled by an Arduino microcontroller with Wi-Fi connectivity. For security, the proposed mailbox captures an image whenever a nearby motion is detected, and opens the mailbox lock when the user has the proper RFID card. Due to the integration of camera, Wi-Fi, RFID reader, and solenoid, the power consumption will be very high and battery or solar power charging might not be possible. Moreover, as there is no sensor used to detect actual mail inside the mailbox, false alarms will be generated if someone moves in front of the mailbox without the intention of checking or stealing mail.

Table 1. Comparison with other works.

	Power Consumption	Solar Charging	Door Locking	Smart Speaker	Wireless Link	Alert
[8]	High	No	No	No	GSM	Email/SMS
[10]	High	No	Yes	No	Wi-Fi	Email
[11]	Medium	Optional	No	No	915 MHz TI radio transceiver	Text, email, or Twitter
[12]	Low	No	No	No	BLE	Smartphone notification only when near the mailbox (not real-time)
[13]	High	—	Yes	No	Wi-Fi	Smartphone Notification
Proposed	Low	Yes	No	Yes	BLE	Smartphone Notification

GSM, Global System for Mobile; BLE, Bluetooth Low Energy; SMS, short message service.

Several startups are planning to make a commercial smart mailbox [11–13]. The work in [11] uses 915 MHz TI radio transceiver for communication, which consumes 45 mA during transmission [14]. In [12], notification to the smartphone is only sent when the user is close to the mailbox—thus, the notification is not real-time. The work in [13] uses Wi-Fi-equipped Raspberry Pi (RPI) in the mailbox. However, Rpi consumes more than 200 mA current during idle state and more than 300 mA current

when Wi-Fi is enabled [15]—thus, making it power-hungry and unsuitable for battery or solar operated use. The Informed Delivery service [16] of the United States Postal Service (USPS) can send emails each morning containing the images of today’s mail. However, this does not provide notification when the mail has arrived. Also, these emails do not contain images of mail from postal services other than USPS.

Compared with other works, the proposed work in this paper contributes to the following points:

- Ultra-low power consumption of the hardware as it uses BLE and event-based low power firmware design.
- The mailbox contains recharging circuitry to charge the battery from a solar panel, thus, no manual charging nor is the wiring of power supply near the mailbox required.
- The proposed mailbox contains a sensor to detect mail in the mailbox, thus, false alarms are not generated if the mailbox door is opened without putting mail inside it.
- The hub in the proposed system connects with the home Wi-Fi and can send a notification to the smartphone to any place in the world in real-time.
- The state-of-the-art smart speaker, Google Home, is interfaced with the system which can speak the status of the mailbox. To date, no related work is found in the literature on interfacing smart-speaker with a mailbox.
- The status of the mailbox can also be viewed from a web browser from any place of the world.

2. Materials and Methods

The overall system architecture of the proposed smart mailbox system is shown in Figure 1. Whenever the door of the mailbox is closed, the electronic device wakes-up from low power sleep mode, senses the existence of mail inside the mailbox using infrared (IR) reflectance sensor, and sends the data to a hub using BLE. The device is powered from a rechargeable battery and it automatically gets charged from a solar panel. The hub—placed inside the home—is wirelessly connected with the home Wi-Fi and it sends a cloud message using the Internet. The smartphone app is notified immediately using push notification. The smartphone app also contains settings screen to configure the hub wirelessly. When a customized question is asked to Google Home, which is connected with the same home Wi-Fi network, the hub casts audio response to Google Home depending upon the status of the mailbox. The hub acts as a web server and the status of the mailbox can also be viewed from a web browser.

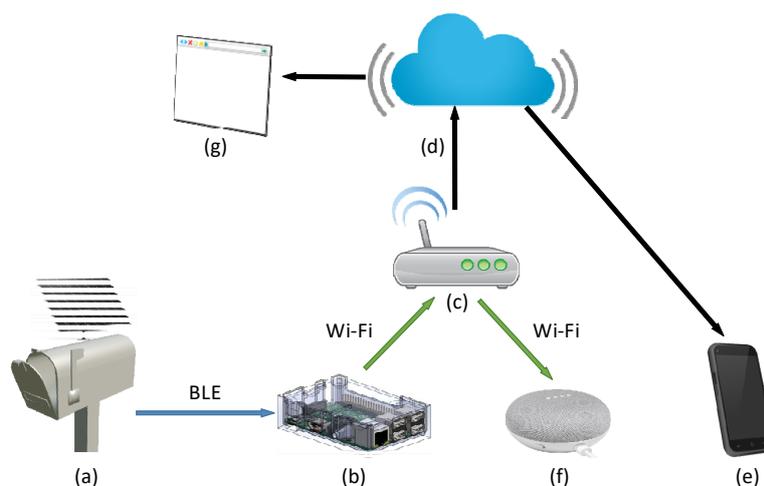


Figure 1. The architecture of the smart mailbox system. The smart mailbox (a), with the electronic device and solar panel, is connected wirelessly with the hub (b) using BLE. The hub is connected with the home router (c) using Wi-Fi. The hub sends cloud message (d) using the Internet to the smartphone app (e) whenever mail is arrived or removed. Google Home (f) is connected with the local Wi-Fi network and can say the status of the mailbox. The status of the mailbox can also be viewed using a web browser (g).

The key design challenges of this IoT system are reducing the power consumption, range of wireless devices, real-time notification, reliability, and smart speaker interfacing. The rest of this section discusses the different components of the system, and how the challenges are addressed and implemented.

2.1. Smart Mailbox

The smart mailbox detects the presence of mail inside it and sends the data to the hub, whenever the door of the mailbox is closed. A microcontroller-based hardware device is designed for this purpose. The key design challenge of the hardware is lowering the power consumption, as it is powered by a battery and recharged using a solar panel. The hardware and firmware of the device are briefly described below.

2.1.1. Hardware

The block diagram of the hardware unit inside the smart mailbox is shown in Figure 2.

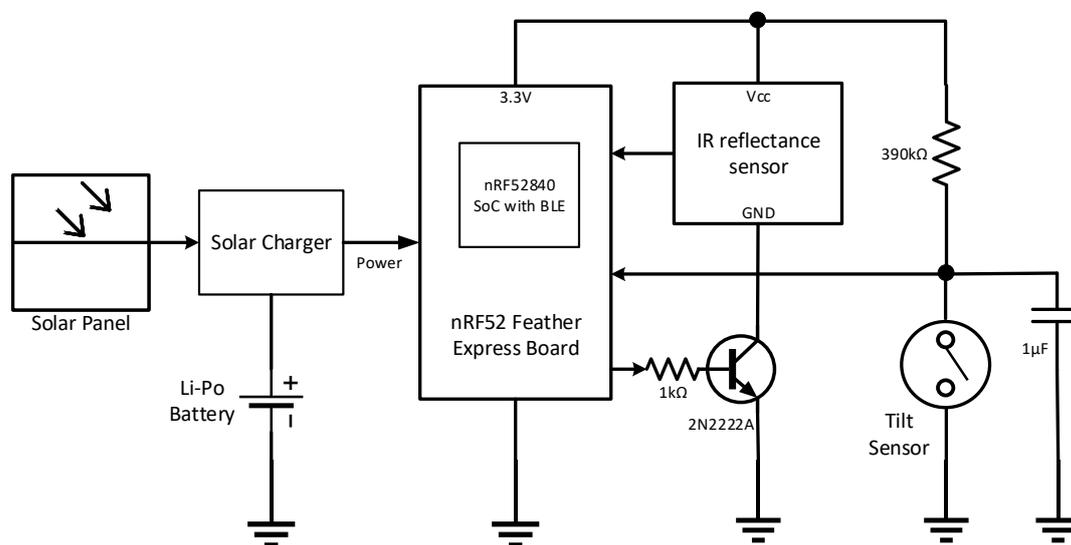


Figure 2. Block diagram of the smart mailbox hardware device.

Nordic nRF52840 [17] is a low-power, small-size system-on-chip (SoC) micro-controller with BLE, and it is used as the processing and wireless communication unit. The micro-controller incorporates an industry-standard ARM Cortex M4F microprocessor, 1 MB in-system programmable flash memory, 256 kB RAM, 2.4 GHz BLE 5 transceiver having output power up to +8 dBm, 6 analog-to-digital converter (ADC) channels, general-purpose input/output (GPIO), Timer, I2C, and many different peripherals. Its low-power sleep modes are appropriate for systems where ultra-low power consumption is required to extend battery life. BLE 5 has a maximum data rate of 48 Mbps and distances up to 300 m. It also introduces a new mode of operation, known as the *long-range* mode, that can transmit data from a distance of 1.6 km [18].

A Feather nRF52840 Express [19] board is used in this project, which consists of the nRF52840 microcontroller. The board has a USB-Serial converter for programming, a connector for 3.7 V lithium polymer (Li-Po) battery, onboard 3.3 V regulator, and a battery charging circuit. The programming and charging circuits only get power when the board is connected with USB and no longer consume power when the battery is connected only. When both battery and USB are connected, the charging circuit starts to charge the battery from USB power. A voltage divider circuit—that is powered from the battery connector—is implemented on the Feather board, and its middle point is connected to an ADC pin of the microcontroller—to monitor the voltage level of the battery.

A 6-watt solar panel [20] is connected with a solar charger module [21] to charge the battery whenever sunlight is available. A 3.7V Li-Po rechargeable battery [22] having a capacity of 2500 mAh is connected with the battery terminal of the solar charger module. The module uses the LT3652 monolithic chip that implements maximum power point tracking (MPPT) method for single-cell Li-Po battery, capable of charging at a rate of maximum 2 A. For this application, the maximum charge current is set to 450 mA.

A tilt sensor switch [23] is used to detect the door position of the mailbox—whether it is closed, having a vertical orientation, or it is open, having a horizontal orientation. When the sensor is upright, a metal ball inside the sensor makes contact with the sensor terminals; when the sensor is tilted, the metal ball disconnects the terminals. The sensor is placed on the mailbox door in an angular orientation so that, whenever the door is closed, the sensor terminals are open, and whenever the door is open, the sensor terminals are closed. The sensor is connected to a GPIO interrupt pin of the microcontroller and the interrupt is configured at the rising event. A 1 μ F capacitor is connected from the MCU pin to the ground to filter out the bounce signal from the sensor. An external pull-up resistor of 390 k Ω —as shown in Figure 2—is used to make the pin high when the sensor terminals are open. Internal pull-up resistors (R_{pu}), having a value of approximately 13 k Ω , are available in the nRF52832 [17]. However, enabling the internal pull-up resistors will cause this branch through the switch to continuously consume $3.3 \div 13k = 0.25$ mA current whenever the sensor terminals are closed. This significant current will quickly reduce battery life. So, a higher value external pull-up resistor of 390 k Ω is used by disabling the internal pull-up resistors. With this solution, the branch consumes only $3.3 \div 390k = 8.46$ μ A current (measured 8.32 μ A in real-time) when the sensor terminals are closed. If a higher value of external resistor such as 1 M Ω is used, then too much voltage drops, and the microcontroller pin does not recognize an interrupt event. That is why a resistor of 390 k Ω external pull-up resistor is used—which is less than 1 M Ω and larger than the internal pull-up of 13 k Ω . The microcontroller input pin sinks insignificant current as the input resistance of the pin is high. When the mailbox door is closed: the sensor terminals are opened, the microcontroller pin goes from low to high and triggers a rising interrupt event.

A miniature IR reflectance sensor, QRE1113 breakout board [24], is used to detect the existence of mail in the mailbox. The sensor is comprised of two parts—an IR emitting light-emitting diode (LED) and an IR sensitive phototransistor. When power is applied to the breakout board, the IR LED of the sensor lights up. The output voltage of the sensor goes from high to low when the IR from the LED is reflected on the phototransistor. The higher the IR light is sensed by the phototransistor, the lower the output voltage of the breakout board. The output pin is connected with an ADC pin of the microcontroller. When mail is put in the mailbox: it covers the sensor, the IR gets reflected by the mail, more IR is sensed by the phototransistor, and output voltage becomes lower. The sensor is cut off from power when it is not used to reduce power consumption. To connect and disconnect the sensor's ground pin from the power supply ground, an NPN transistor [25] is used. A GPIO pin of the microcontroller, as shown in Figure 2, controls the base of the transistor. The base current flows only for a few milliseconds during stabilization and sensing, and then the current is made zero by turning the GPIO pin to low, to cut off the power from the sensor.

2.1.2. Firmware

The electronic device, placed in the smart mailbox, sends data to the hub using BLE. The mailbox device acts as *peripheral* and the hub acts as *central* [26]. A feature-rich BLE 5 protocol stack for the nRF52840 SoC—referred to as *SoftDevice*—is implemented that contains complete stack with Generic Access Profile (GAP), Generic Attribute Profile (GATT), Link and other layers [27]. The BLE protocol supports custom data transmission either using advertisements in a non-connected state or using services in a connected state. To transmit data in a connected state, the firmware must use a service and the service must contain one or more characteristics [26]. Services are used to break data in groups and each group contains specific chunks of data called characteristics. A service can have

one or more characteristics, and each service distinguishes itself from other services using a unique numeric ID called a Universally Unique ID (UUID), which can be either 16-bit (for officially adopted BLE Services) or 128-bit (for custom services). Like services, each characteristic distinguishes itself via a pre-defined UUID. Custom service is implemented by the firmware in the GATT profile. Under this service, a custom characteristic is created. The data length of the characteristic is 2 bytes—one byte to indicate the existence of mail and another byte for battery level. The *notify* property of the characteristic is set so that a call back function in the hub is called automatically whenever the data in the characteristic is updated.

To make a connection with the hub, the device first advertises with its GAP data. Then the hub gets connected with the device and remains at the connected state. The BLE transmit power of the device is set to its maximum value of +8 dBm. Once connected, the device stops advertising and it becomes ready for data transmission. If the BLE is disconnected—due to low battery or range issues—a BLE disconnected call back function is called in the hub and the device starts advertising again for reconnection.

The firmware uses a real-time operating system (RTOS)—referred to as *FreeRTOS* [28]. Instead of always running codes inside a loop, the firmware executes call back functions on events to save power. The firmware waits for an event to happen to wake up and then again goes to low power sleep mode. Whenever the mailbox door is closed, the tilt sensor becomes open, the *door_closed* interrupt is triggered, and the microcontroller wakes up. The interrupt service routine (ISR) then starts a timer and also applies power to the IR reflectance sensor by making the GPIO pin high, which is connected with the base of the transistor. During the delay from the timer, the mailbox is expected to become stable from motion. After the delay, the timer ISR is called. In this ISR, the timer is first stopped. Then the ADC peripheral is enabled, IR reflectance sensor data and battery levels are read, then the ADC peripheral is disabled. The IR reflectance sensor data is compared with a threshold to determine whether mail is in the mailbox. The existence of mail and the battery level data are then written on the BLE *characteristics*—causing the execution of a call-back function in the hub to receive the data. Lastly, the power is cut off from the IR reflectance sensor by making the GPIO pin low, which controls the transistor. Then the program goes to sleep mode and waits for another door close event to happen. The actions executed by the device after the door of the mailbox is closed is shown in Figure 3.

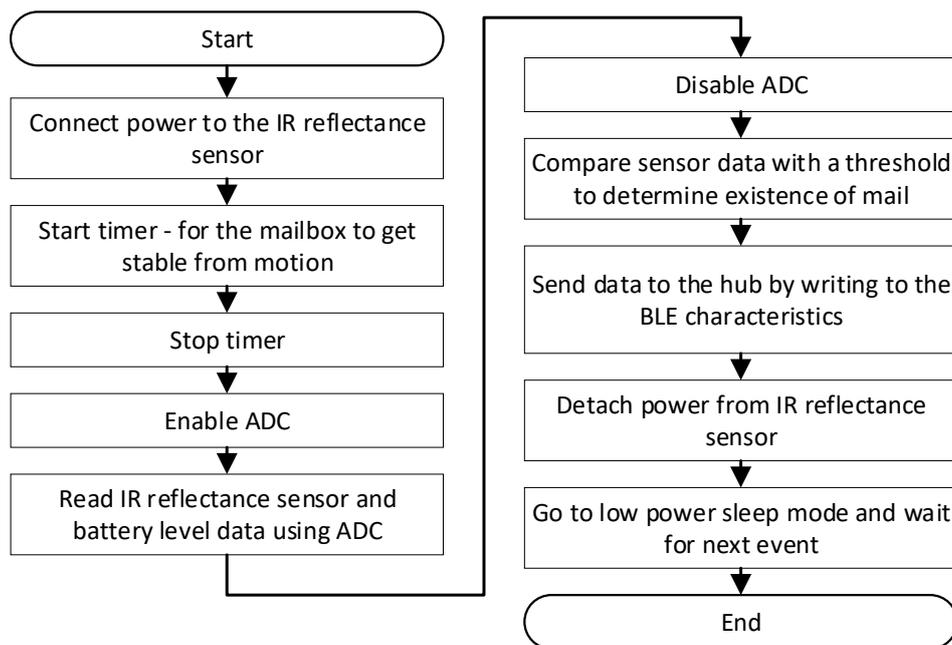


Figure 3. Flowchart of actions after the door of the mailbox is closed. IR, infrared; ADC, analog-to-digital.

In the firmware, the following configurations are done to reduce the power consumption:

- Instead of using the low dropout (LDO) regulator, the DC/DC converter of the microcontroller is used to decrease power consumption [29].
- The floating-point unit (FPU) consumes significant power. So, the program was written avoiding any floating-point number. The exception flags and the pending FPU interrupts were cleared to take the microcontroller in low power mode [30].
- The RTOS uses a tick interrupt to measure time. The increase of the tick frequency means that the timer can measure time to a higher resolution, and vice versa. The RTOS scheduler also shares processor time between tasks of the same priority by switching between the tasks during each RTOS tick. The tick rate frequency affects the time slice given to each task [28]. To reduce power consumption, the frequency of the RTOS tick interrupt (referred to as configTICK_RATE_HZ) was decreased from 1024 Hz to 64 Hz and it reduced 110 μ A current. Further decreasing the tick frequency causes problems in the ADC peripheral. The system clock of the microcontroller (referred to as SystemCoreClock) is 64 MHz and it is not changed. Thus, the microcontroller does not sacrifice its performance and executes instructions at a high speed.

2.2. The Hub

The hub receives data from the mailbox using BLE and sends a cloud message using the home Wi-Fi to the smartphone app. The smart speaker and web interface also connects with the hub using the home router. The hub can be placed inside the home, within the range of the BLE and home Wi-Fi signal. The hardware and app parts of the hub are described below.

2.2.1. Hardware

A single-board computer, Raspberry Pi (RPi) v3 [31], is used as the main processing unit. It contains a 64-bit quad-core ARMv8 microprocessor running at 1.2 GHz, 1 GB of RAM, a micro SD card slot supporting up to 32 GB, on-board BLE, Wi-Fi module, and other built-in hardware peripherals. A 110 V AC to 5.1 V DC adapter [32] is used as the power supply for the RPi board.

2.2.2. Network Configuration

The hub must be accessed from outside of the home network using the Internet, to interface it with a smart speaker, get mailbox status from a web browser, and to update its settings from the smartphone. So, the hub is made a Hypertext Transfer Protocol (HTTP) server, which can receive and send data from/to the Internet using the transmission control protocol/Internet protocol TCP/IP. To access the server, it should have a fixed address and a port number. The local (or private) IP of the hub is dynamically assigned by the dynamic host configuration protocol (DHCP) server of the router; thus, the IP address of the hub may change on a new connection depending on how many devices are currently connected with the router. To solve this, the local IP of the hub is made static [33] and port forwarding [34] is configured in the router. The port forwarding mechanism will transmit the data packet, which arrives at the home router from the Internet, to the hub's HTTP server. A friendly and memorable name for the home router's public IP address is assigned using a free dynamic domain name system (DDNS) [35] server. The public IP of the home router—assigned by the Internet service provider (ISP)—does not change frequently, but may change after months or years. To solve this problem, the router is configured by entering the DDNS provider's login information. This ensures that the public IP address in the DDNS server is automatically updated if it changes [36].

Note that the setting of the hub IP as static can also be done using a python program without manually configuring the router. The hub firmware may contain a function to write in the `/etc/dhcpd.conf` file as described in [37] to set its IP as static. Other router configurations can also be done from the python program using libraries such as `rosraco` [38], which will eliminate the need for configuring the router manually. We plan to automate the router configuration in the future.

2.2.3. Firmware

Raspberry Pi OS (previously called *Raspbian*) [31]—a Debian-based Linux operating system—is installed on a 16 GB SD card of the RPi board. The firmware is developed in Python language and the necessary packages such as *pygatt*, *pyfcm*, *pychromecast*, *http.server* are installed.

At startup, the hub gets connected to the home router using Wi-Fi and thus can access the Internet. The last saved status of the mailbox and some settings—such as the presence of mail, battery level, Google Home Internet Protocol (IP) address, and BLE media access control (MAC) address—are loaded from the *settings.dat* file. The hub then connects with the mailbox using BLE and subscribes to the custom characteristic's UUID.

Whenever the door of the mailbox is closed, the mailbox's status is sent to the hub using BLE, and a call back function—*BLE data arrival*—is called in the hub. The actions executed after the hub receives the BLE data are shown in Figure 4. In the *BLE data arrival* function, the status of the mailbox—such as the presence of mail, battery level, and BLE connection status—is saved in *settings.dat* file and is sent to the user's smartphone using Firebase Cloud Messaging (FCM) [39]. FCM is a cross-platform messaging solution that reliably sends messages at no cost using the Internet. Using FCM, a smartphone app can be immediately notified whenever new data is available to sync. A message can carry a payload of a maximum of 4 KB to a client app. The message is sent using a *Server key* that is generated from the cloud server where the smartphone app is registered. If the BLE connection with the mailbox is disconnected—due to low battery or range issues—a *BLE disconnected* call back function is called in the hub. The function then updates the smartphone about the BLE disconnected status using FCM.

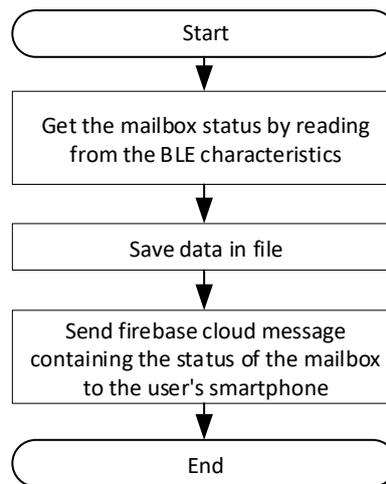


Figure 4. Flowchart of actions after the hub receives the BLE data.

An HTTP server is implemented in the firmware. Whenever an HTTP GET request to the server is made, a call back function is called in the firmware. The server of the hub needs to be accessed using the Internet: to configure its settings from a smartphone—as the hub does not have any display or keyboard, to get the mailbox status in the smartphone, to get the mailbox status from a web browser, and to say the status of the mailbox using a smart speaker. The Uniform Resource Locator (URL) of the GET requests contains the words listed below. The hub executes the following actions upon receiving these requests:

- *Settings*: This request is made by the smartphone app. The URL of this request consists of the MAC address of the smart mailbox's BLE and the local IP address of Google Home. Once the hub gets this request: the BLE MAC and the Google Home IP address are parsed from the URL, saved in the *settings.dat* file, reconnect with mailbox using the new BLE MAC, and send the mailbox status—the presence of mail, battery level, and BLE connection status—to the smartphone app using FCM.

- *Status*: This request is made by the smartphone app to get the current status of the mailbox. When this request is received, the hub sends the mailbox status to the smartphone using FCM.
- *Web*: This request can be made by the user from any web browser to get the status of the mailbox. The hub responds to this request by sending an HTML message—containing whether mail is present in the mailbox and the battery charge level—to the web browser.
- *Voice*: This request is made by the *If This Then That (IFTTT)* [40] web service to interface with the Google Home smart speaker. This request is made whenever the user asks Google Home to “check the mailbox.” Upon receiving this request: the hub creates a new thread for casting audio to Google Home, connects with Google Home using its IP, and casts audio “you have mail in your mailbox” or “your mailbox is empty”—depending upon the status of the mailbox. These two audio mp3 files are uploaded in Firebase Cloud Storage [41] and the hub access these files using a secure URL. This audio casting is done using a python library named *pychromecast* [42].

2.3. Smartphone App

The smartphone app is developed for the Android platform. When the app starts, it sends an HTTP GET request to the hub’s server using the *status* word in the URL. The hub then sends the status of the mailbox using FCM to the smartphone. The first screen of the app shows the status of the mail—“you have mail” or “mailbox is empty;” the percentage of battery charge level; date and time of the last data update; its connection status with the hub using the Internet; and the hub’s connection status with the mailbox using the BLE. It also contains a button to reconnect with the mailbox using BLE and it is enabled when the connection of the hub with the mailbox is lost.

The app contains a settings menu for configuring notifications, the hub’s public IP, Google Home IP, and mailbox’s BLE MAC. The user can enable or disable notifications: at the event of a new mail, at the event of when the mailbox is empty, or both. It might be necessary to get a notification when the mail is taken out and the mailbox becomes empty to detect mail theft. To make an HTTP request to the hub from the smartphone app, the public IP, or the friendly name of the hub server is required. If the smartphone is connected with the same Wi-Fi network of the hub, then the public IP of the Wi-Fi network can be autofilled by pressing a button that sends an HTTP request to <https://ipecho.net/plain> and it responds the IP from where the request was made. Alternatively, the public IP can also be manually entered by looking it up from the whatsmyip.com website. To audio cast the status of the mail to the smart speaker, the hub must know the Google Home IP. The user enters the Google Home IP in the app. Google Home IP can be found from the *Home* app [43]. The hub must know the BLE MAC for connecting with the mailbox. To get the BLE MAC of the mailbox, the user presses the ‘Start Scan’ button to find the nearby BLE devices that are advertising. The nearby BLE devices are added in a list-view box and the user can select the required MAC address from the list. Once the user exits from the settings menu by pressing the back button, the app saves the data in *settings.dat* file and sends an HTTP GET request to the hub using the *settings* word, the BLE MAC, and the Google Home IP in the URL.

The first screen of the app is updated whenever a new FCM message arrives. The flowchart in Figure 5 shows the actions taken by the smartphone whenever the FCM message is received. In this way, the data is synchronized between the hub in the home and the user’s smartphone—at whatever place the user may be in the world. The app is added and registered in the FCM for receiving the push notification. A service, named *FirebaseMessaging*, runs at the background of this app. When it receives a push notification message from the FCM, a call back function is called and the app saves the mailbox status—such as the presence of mail, battery level, and BLE connection status—with the current date/time information in the *status.dat* file. The app then generates a smartphone notification if new mail arrived or the mail is removed, depending on the notification settings and last status. The first screen of the app is then updated to show the current status of the mailbox and battery level. If the hub’s BLE connection with the mailbox is lost, then the connection status of the mailbox is shown as “disconnected;” and the presence of mail and battery level is shown as *unknown*.

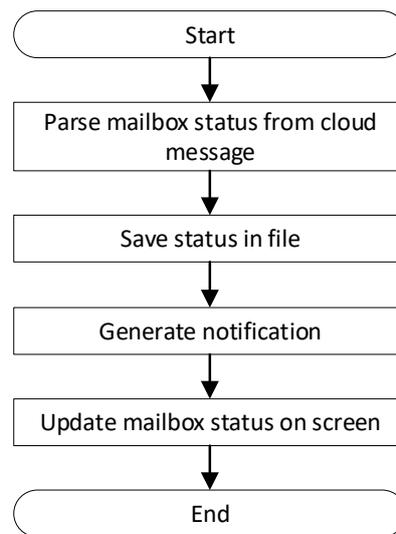


Figure 5. Flowchart of actions when the Firebase Cloud Messaging (FCM) message is received.

If the hub server is down, then any HTTP request sent from the smartphone will result an error message. In this case, the connection status of the hub is shown as “disconnected” and the presence of mail, battery level, and BLE connection status with the mailbox are shown as *unknown* in the first screen of the app.

2.4. Smart Speaker Interface

A smart speaker, Google Home, is interfaced with the system. When the user asks Google Home: “Ok Google, check physical mailbox”, it answers “you have mail in your mailbox” or “your mailbox is empty”, depending upon the status of the mailbox. This customized question is implemented using an *If This Then That (IFTTT)* [40] applet. Simple IFTTT applet can recognize customized questions and can answer a static response, but it cannot choose from a set of answers based on sensor input. To solve this problem, the action of the applet is made to be a web request using Webhooks [44]. IFTTT makes an HTTP GET request to the hub’s friendly name and port number with the word *voice* in the URL. After this request is received, the hub first connects with the Google Home using its local IP. The Google Home IP is made static by configuring the router. Then, depending upon the status of the mailbox, the hub casts the audio—“you have mail in your mailbox” or “your mailbox is empty”—to the smart speaker.

2.5. Web Interface

The status of the mailbox—whether empty or having mail—can be read using a web browser from any place of the world. An *index.html* file was placed in the same folder of the hub’s python code—running the HTTP server. The website can be accessed using the public IP or the friendly name of the hub using any web browser from any place of the world. The website is written using HTML and it contains a hyperlink—“Get status”. When the user clicks it, it makes an HTTP GET request to the hub’s friendly name and port number with the word *web* in the URL. The hub then responds to this request by sending an HTML message—containing whether mail is present in the mailbox and the battery charge level—to the web browser.

3. Results

A prototype of the proposed smart mailbox, the hub, smartphone app, Google Home interface, and web interface has been developed and tested successfully. A photograph of the smart mailbox—with the electronic device inside it—is shown in Figure 6. The current consumption of the device at a different

state was measured using an ammeter and shown in Table 2. The BLE of the device will advertise only when it needs to be discovered by the hub. Once discovered and connected, the device consumes 780 μ A current. The mailbox will be at this low power idle state most of the time. When the door is opened and then closed, the device wakes up from sleep mode, senses the presence of the mail and transmits data for a short amount of time for few milliseconds, and again goes to sleep mode. The prototype uses a Li-Po rechargeable battery having a capacity of 2500 mAh and it is connected with a solar panel through a solar charger circuit. Whenever sunlight is available, the battery will be charging. Assuming no solar charging is available, the battery life of the device is approximately $2500 \div 0.78 = 3205.12$ h or 133 days. So, the smart mailbox will work even if the weather is cloudy for many days. As the battery will be charging during the daytime, the proposed mailbox will not run out of power and no manual charging will be required.

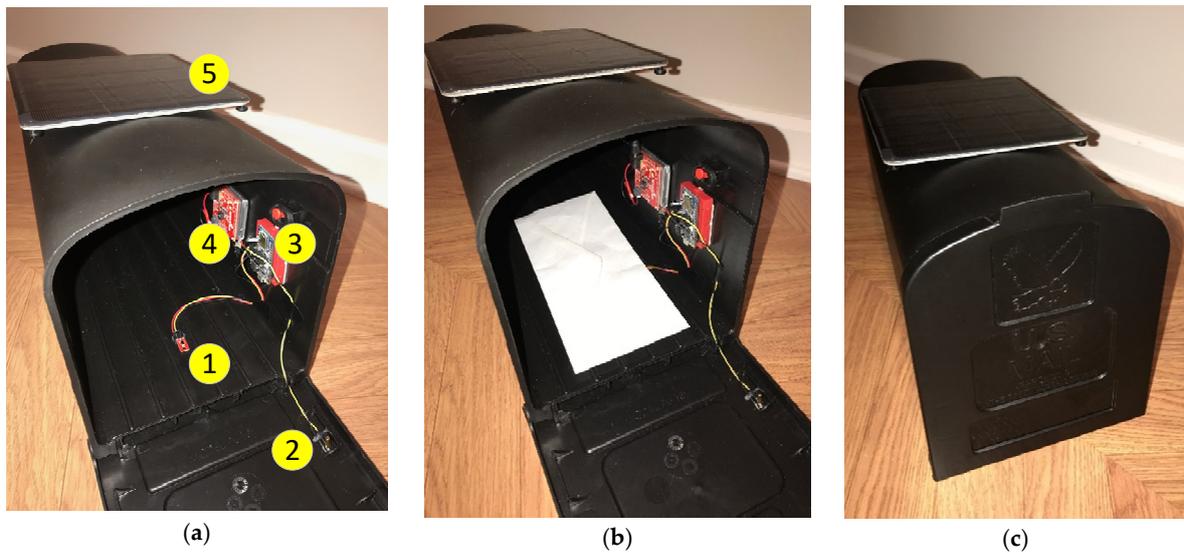


Figure 6. Photograph of the smart mailbox: (a) electronic device placed inside the mailbox—(1) IR reflectance sensor, (2) tilt sensor, (3) system-on-chip (SoC) microcontroller with BLE, (4) solar charger and Li-Po rechargeable battery, and (5) solar panel; (b) mail in the mailbox—covering the IR reflectance sensor; (c) when the door is closed, the tilt sensor causes an interrupt to the microcontroller.

Table 2. Measured current consumption of the device at different states.

Device State	Current Consumption (μ A)
BLE advertising	630–900
BLE connected and idle	780
Sensing and transmitting	22,300

The photograph of the hub—that is placed inside the house—is shown in Figure 7. The screenshots of the smartphone app are shown in Figure 8. The first screen of the app shows the status of the mail; the percentage of battery charge level; date and time of the last data update; its connection status with the hub; the hub’s connection status with the mailbox using the BLE; and a button to reconnect the hub with the mailbox if the connection is lost. The app generates notifications when mail arrives, or mail is removed. The status of the mailbox can be known by asking Google Home smart speaker and also using a web browser—as shown in Figure 9.

The smart mailbox system was tested in the real world by putting the mailbox around 100 feet away, across a 30-foot street, from the hub. The hub was placed inside the home about 3 feet high from the ground. When mail was put and the door of the mailbox was closed, the mailbox status was updated immediately on the smartphone app. The smart speaker could answer the status of the

mailbox and the status could also be read using a web browser. We also took the smartphone out of the range of home Wi-Fi and it was connected with the Internet using the cellular network. The smartphone successfully received notification in this scenario also.

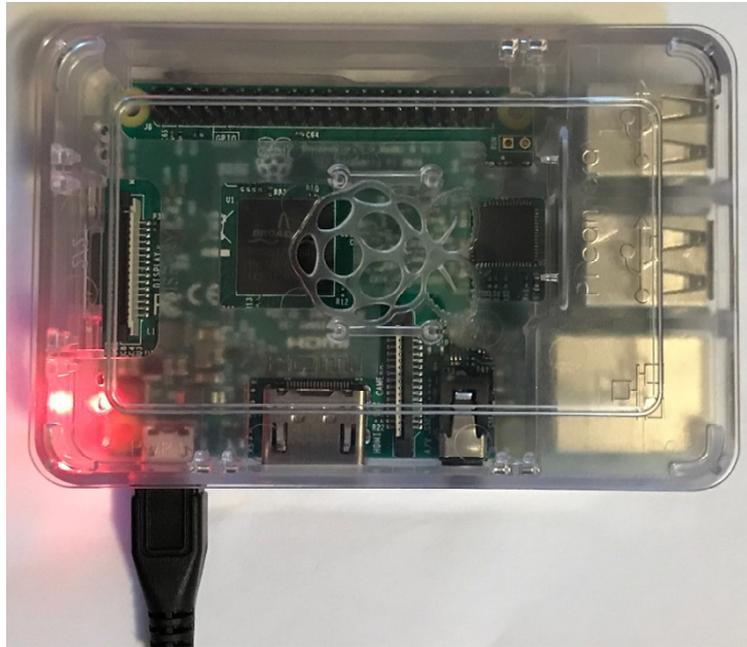


Figure 7. Photograph of the hub—implemented in a Raspberry Pi board.

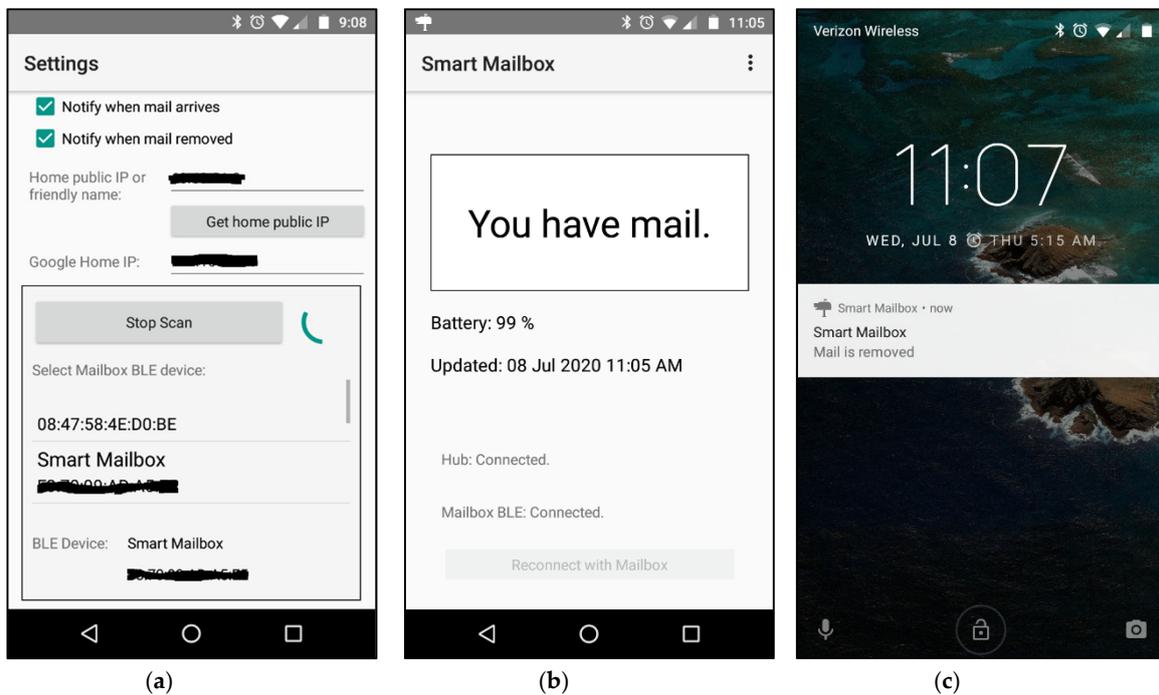


Figure 8. Screenshots of the smartphone app: (a) Settings window for configuring notifications, the hub’s public IP or friendly name, Google Home local IP, and the mailbox’s BLE MAC; (b) The first screen of the app shows the status of the mail; the percentage of battery charge level; date and time of the last data update; its connection status with the hub; and the hub’s connection status with the mailbox using the BLE. It also contains a button to reconnect the hub with the mailbox if the connection is lost; (c) Notification when the mail is removed.

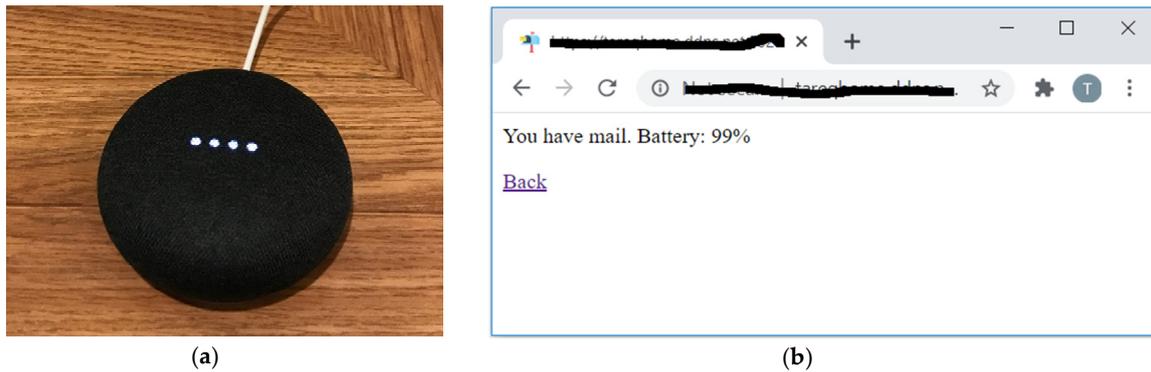


Figure 9. (a) When the user asks Google Home: “Ok Google, check physical mailbox”, it answers “you have mail in your mailbox” or “your mailbox is empty”, depending upon the status of the mailbox; (b) the mailbox status can be read using a web browser.

4. Discussion

In this project, the IR reflectance sensor is used to detect the presence of mail or object inside the mailbox. We have put papers and plastic objects of different colors—such as white, blue, red, green, yellow, black—on the sensor, and checked whether the sensor output voltage goes lower than a threshold or not. The output pin of the sensor is connected with the ADC of the microcontroller as described in Section 2.1. We found that the output voltage of the sensor goes significantly low for all colors (except black) and thus it can detect the colored mail. However, if the object is black, then the output voltage of the sensor does not become significantly low because IR rays do not reflect from the black surface. From these observations, we concluded that different colored mail can be detected by the sensor except it is black. In this design, one IR reflectance sensor is used and the mail may be put at the back of the mailbox—not covering the sensor. To ensure that the sensor is covered by mail, more than one sensor can be interfaced and can be placed in the mailbox every after a few inches apart.

We made our first mailbox prototype with Nordic nRF52832 SoC microcontroller [45]—where the maximum BLE transmit power can be set to +4 dBm. During testing, we found that it consumes lower power, but the range of BLE transmission is only around 30 feet in open space. As the mailbox can be placed farther away than 30 feet from the home, we switched to the new Nordic nRF52840 SoC microcontroller [17], where the BLE output transmit power can be increased up to +8 dBm. With this microcontroller, we got around 100 feet of range. The range of BLE transmission may vary depending upon the obstacles—such as trees and walls—in the path, weather condition, etc. If there exist large obstacles from the mailbox to the hub that largely attenuates the BLE signal, then the system may face a frequent disconnection event. One way to solve this problem is to use one or more solar-powered intermediate microcontroller nodes and use BLE mesh networking [46]. Note that the nRF52840 has *long-range* BLE support that can transmit data from a distance of 1.6 km [18]. However, the RPi in the hub still does not have software support for this long-range feature. We plan to use this feature when RPi supports long-range BLE.

The proposed work uses the Nordic nRF52840, which contains an ARM microcontroller and BLE transceiver in a single chip. This chip is designed for low power applications in mind, even though it contains a powerful ARM processor. Its low energy consumption is achieved using a sophisticated on-chip adaptive power management system [17]. Another option could be to use Texas Instruments CC2541 that contains an 8051 microcontroller with BLE in a single chip. However, it has a short BLE range as the BLE output power can be a maximum of 0 dBm. The Nordic nRF52840 has a longer BLE range as the output power could be 8 dBm. Moreover, a better breakout board [18], library, and compiler support are available for the Nordic microcontroller. Considering these advantages, the nRF52840 SoC is used in this application.

The SoC microcontrollers with BLE, such as Nordic nRF52840 and Texas Instruments CC2541, need an operating system to handle low-level BLE stacks and for power management. For instance, in the CC2541, the BLE protocol stack, the profiles, and all applications are all built around the Operating System Abstraction Layer (OSAL). The OSAL is a control loop that allows the software to the execution of events according to priority. In Nordic nRF52840, the BLE library [47] is built using the FreeRTOS [30]. FreeRTOS is a small kernel that provides methods for multiple tasks with priorities, mutexes, semaphores, and software timers. A tick-less mode is provided for low power applications and this part is used for taking the SoC in sleep mode. Thus, using a simple operating system is a requirement for these SoC chips. One way to avoid the RTOS is to use the UART BLE modules [48] that can be operated using AT commands. However, this approach will need another chip as the microcontroller and will not be power efficient.

As soon as the mailbox door is closed, an interrupt happens, the device wakes up from sleep mode, detects the presence of mail, sends the status to the hub, and then the device goes to sleep mode. If the door is opened and then closed again, the same process repeats. If in a country where mail is regular, every two or three days, then this pattern might be used for further power optimization. One implementation could be that after receiving mail, the device may go to a complete power-down mode for several hours and then come back to a normal sleep mode using an external timer. This would require powering the external timer while the device is in power-down mode. Further investigation is required whether this optimizes the power consumption.

Regarding timing evaluation, the proposed system is a real-time system. As soon as the door of the mailbox is opened and then closed, an interrupt is generated, and the mailbox status is sent to the hub using BLE. Upon receiving the BLE data, the hub sends a push notification to the smartphone app using FCM and the user is notified. These events happen in less than a second and the user is notified immediately.

The proposed system notifies the status of the mailbox reliably almost all the time. As mentioned earlier, if the color of the package is black or the package is not placed on top of the IR reflectance sensor, then the device will not detect the mail and it will be a false negative. Most packages and envelopes are not fully black, and the sensor should be able to detect most mail. One way to overcome this limitation is to use an ultrasonic distance sensor instead of an IR sensor; however, this will slightly increase the power consumption. To make sure the sensor is covered by the mail, several sensors can be placed in the mailbox every after a few inches apart. We plan to implement this in the future. During the evaluation, no false-positive notification was generated by the system.

It is possible to broadcast the mailbox status data in the BLE advertisement packet without connecting to a specific device. This might help to further reduce power consumption. However, we have chosen to transmit data in the connected mode because broadcasting mailbox status in advertising packets has no privacy. Anyone in a neighborhood can monitor the mailbox status change wirelessly using a BLE scanner app and it might increase mail theft. It might be possible to manually encrypt the data. However, the change in the status will also change the encrypted data and can notify the hacker. Data transmission in connected mode will not allow other people to monitor the mailbox status change wirelessly.

Privacy and security are a concern for IoT devices. We plan to make the system more secure in the future by utilizing *POST* requests instead of *GET* requests in the hub's server and also encrypting the data. In the proposed firebase cloud messaging system, the smartphone app must be subscribed to a unique topic [49] that can be a secret key and the hub must know a secret server key (consisting of more than 150 characters) to send the notifications. Though these give some protection, more work on IoT privacy is required to make the system more secure. Electronic locking system controlled by a smartphone can be implemented to secure the mail from theft; however, operating the lock's solenoid will consume large power and might be a challenge for solar-powered devices.

A warning can be generated when the mailbox is getting full. One possible implementation is to use a proximity sensor that can be placed at the ceiling of the mailbox to measure the distance from the

mail stack. If the distance gets lower than a threshold, then the mailbox is getting full and a notification to empty the mailbox can be sent to the smartphone. We plan to implement this functionality in the future.

5. Conclusions

In this paper, an IoT connected smart mailbox system is proposed, developed, and tested. The electronics in the mailbox consume ultra-low power as it uses BLE and the battery is charged using a solar panel. The hub in the proposed system connects with the home Wi-Fi and can send a notification to the user's smartphone to any place in the world in real-time when mail arrives. Moreover, the state-of-the-art smart speaker, Google Home, is interfaced with the system, which can speak the status of the mailbox. The mailbox status can also be viewed from a web browser from any place of the world.

Author Contributions: Conceptualization, methodology, software, validation, analysis, investigation, resources, writing—original draft preparation, writing—review and editing, visualization, supervision, project administration, funding acquisition: T.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the GameAbove Faculty First award of Eastern Michigan University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Paul, C.; Ganesh, A.; Sunitha, C. An overview of IoT based smart homes. In Proceedings of the 2018 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 19–20 January 2018; pp. 43–46.
2. Quinones, J.; Lofton, A.; Khan, T.H. A Smart Door with Web Controllable Door Motion and Window Transparency. *Int. J. Eng. Sci. Invent.* **2020**, *9*, 6–12.
3. Oliver, K.; Herter, M.; Khan, T.H. An Architecture of a Touchscreen Operated Window Blinds for Smart Home. *Int. J. Eng. Sci. Invent.* **2020**, *9*, 34–39.
4. Khan, T. A Low Power IoT-Connected Smart Canister System Creating Automatic Shopping List. *J. Sens. Actuator Netw.* **2019**, *8*, 38. [[CrossRef](#)]
5. Khan, T. A Wi-Fi based Architecture of a Smart Home Controlled by Smartphone and Wall Display IoT Device. *Adv. Sci. Technol. Eng. Syst. J.* **2018**, *3*, 180–184. [[CrossRef](#)]
6. Chen, J. Smart Home. Available online: <https://www.investopedia.com/terms/s/smart-home.asp> (accessed on 22 June 2020).
7. Newport, F.; Ander, S. Four in 10 Americans Look Forward to Checking Mail. Available online: <https://news.gallup.com/poll/182261/four-americans-look-forward-checking-mail.aspx> (accessed on 22 June 2020).
8. Subramaniam, S.K.A.; Husin, S.H.B.; Yusop, Y.B.; Bin Hamidon, A.H. Real time mailbox alert system via SMS or email. In Proceedings of the 2007 Asia-Pacific Conference on Applied Electromagnetics, Melaka, Malaysia, 4–6 December 2007; pp. 1–4.
9. Pandya, V.; Shukla, D. GSM Modem Based Data Acquisition System. *Int. J. Comput. Eng. Res.* **2012**, *2*, 1662–1667.
10. Tew, J.R.; Ray, L. ADDSMART: Address digitization and smart mailbox with RFID technology. In Proceedings of the 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 20–22 October 2016; pp. 1–6.
11. The Gate Smart Mailbox. Available online: <https://newatlas.com/gate-smart-mailbox/37408/> (accessed on 22 June 2020).
12. Postifier. Available online: <https://newatlas.com/postifier-postbox-notifications/28373/> (accessed on 22 June 2020).
13. Mr. Postman. Available online: <https://www.engadget.com/2013-11-08-mr-postman-hands-on.html> (accessed on 22 June 2020).
14. CC1120 High-Performance RF Transceiver for Narrowband Systems. Available online: <https://www.ti.com/lit/ds/symlink/cc1120.pdf> (accessed on 23 June 2020).
15. Raspberry Pi Power Consumption Data. Available online: <https://www.raspberrypi-spy.co.uk/2018/11/raspberry-pi-power-consumption-data> (accessed on 23 June 2020).

16. Informed Delivery by USPS. Available online: <https://informedelivery.usps.com/box/pages/intro/start.action> (accessed on 22 June 2020).
17. Nordic nRF52840 SoC. Available online: <https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840> (accessed on 30 June 2020).
18. Bhargava, M. *IoT Projects with Bluetooth Low Energy*; Packt Publishing: Birmingham, UK, 2017; ISBN 9781788399449.
19. Adafruit Feather nRF52840 Express. Available online: <https://www.adafruit.com/product/4062> (accessed on 30 June 2020).
20. Solar Panel-6W. Available online: <https://www.sparkfun.com/products/13783> (accessed on 25 June 2020).
21. MPPT Solar Charger. Available online: <https://www.sparkfun.com/products/12885> (accessed on 25 June 2020).
22. Lithium Ion Polymer Battery—3.7 v 2500 mAh. Available online: <https://www.adafruit.com/product/328> (accessed on 25 June 2020).
23. Tilt Sensor. Available online: <https://www.sparkfun.com/products/10289> (accessed on 25 June 2020).
24. IR Reflectance Sensor Breakout. Available online: <https://www.sparkfun.com/products/9453> (accessed on 25 June 2020).
25. NPN Bipolar Transistors (PN2222). Available online: <https://www.adafruit.com/product/756> (accessed on 30 June 2020).
26. Davidson, R.; Townsend, K.; Wang, C.; Cufí, C. *Getting Started with Bluetooth Low Energy Tools and Techniques for Low-Power Networking*; O'Reilly Media: Newton, MA, USA, 2014.
27. S140 SoftDevice. Available online: <https://www.nordicsemi.com/Software-and-tools/Software/S140> (accessed on 30 June 2020).
28. Free RTOS Customization. Available online: <https://www.freertos.org/a00110.html> (accessed on 1 July 2020).
29. Optimizing Power on nRF52 Designs. Available online: <https://devzone.nordicsemi.com/nordic/b/blog/posts/optimizing-power-on-nrf52-designs> (accessed on 1 July 2020).
30. Floating Point Unit (FPU) of nrf52. Available online: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.0.0%2Fhardware_driver_fpu.html&cp=4_0_9_2_4 (accessed on 1 July 2020).
31. Raspberry Pi. Available online: <https://www.raspberrypi.org> (accessed on 1 July 2020).
32. DC Power Supply. Available online: <https://www.sparkfun.com/products/13831> (accessed on 1 July 2020).
33. How to Give Your Raspberry Pi a Static IP Address. Available online: <https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-give-your-raspberry-pi-a-static-ip-address-update> (accessed on 1 July 2020).
34. Yatritrivedi; Fitzpatrick, J. How to Forward Ports on Your Router. Available online: <https://www.howtogeek.com/66214/how-to-forward-ports-on-your-router/> (accessed on 1 July 2020).
35. Remote Access with Dynamic DNS. Available online: <https://www.noip.com/remote-access> (accessed on 1 July 2020).
36. Fitzpatrick, J. How To Easily Access Your Home Network From Anywhere With Dynamic DNS. Available online: <https://www.howtogeek.com/66438/how-to-easily-access-your-home-network-from-anywhere-with-ddns/> (accessed on 1 July 2020).
37. How to Set A Raspberry Pi with a Static ip Address? Available online: <https://www.ionos.com/digitalguide/server/configuration/provide-raspberry-pi-with-a-static-ip-address/> (accessed on 28 July 2020).
38. Library for Managing Home Routers. Available online: <https://pypi.org/project/roscraco/> (accessed on 28 July 2020).
39. Firebase Cloud Messaging. Available online: <https://firebase.google.com/docs/cloud-messaging> (accessed on 1 July 2020).
40. If This Then That (IFTTT). Available online: <https://ifttt.com/> (accessed on 2 July 2020).
41. Cloud Storage for Firebase. Available online: <https://firebase.google.com/docs/storage> (accessed on 2 July 2020).
42. Library for the Google Chromecast. Available online: <https://github.com/home-assistant-libs/pychromecast> (accessed on 28 July 2020).
43. Google Home App. Available online: https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app&hl=en_US (accessed on 28 July 2020).
44. Webhooks. Available online: https://ifttt.com/maker_webhooks (accessed on 7 July 2020).
45. nRF52832 SoC. Available online: <https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52832> (accessed on 10 July 2020).

46. Introducing Bluetooth Mesh Networking. Available online: <https://www.bluetooth.com/blog/introducing-bluetooth-mesh-networking> (accessed on 26 July 2020).
47. Arduino Core for Adafruit Bluefruit nRF52 Boards. Available online: https://github.com/adafruit/Adafruit_nRF52_Arduino (accessed on 27 July 2020).
48. Adafruit Bluefruit LE UART Friend—Bluetooth Low Energy (BLE). Available online: <https://www.adafruit.com/product/2479> (accessed on 27 July 2020).
49. Topic Messaging on Android. Available online: <https://firebase.google.com/docs/cloud-messaging/android/topic-messaging> (accessed on 27 July 2020).



© 2020 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).