Journal of
Sensor and Actuator Networks

MDPI

Article

A Low Power IoT-Connected Smart Canister System Creating Automatic Shopping List

Tareq Khan

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

Received: 7 June 2019; Accepted: 2 July 2019; Published: 8 July 2019

Abstract: One of the most common forgotten things of adults is that they go to the shops and completely forget what they went for. The solution to this problem is to carry a shopping list. In this project, a novel Internet of Things (IoT)-connected smart canister system is developed, which automatically senses the item quantity in the canisters using proximity sensor, sends the data to a hub using Bluetooth Low Energy, and then the hub sends a cloud message to the consumer's smartphone app using the Internet. The hub and the smartphone app display the item quantities and automatically add the items that are about to finish to a digital shopping list. The automatic generation of the shopping list removes the burden of manually checking each item before going to the shops and gives peace of mind to the consumers. A prototype of the proposed system with three canister devices, one hub, and the smartphone app is developed and tested successfully. The canister device consumes ultra-low power and has a battery life of more than a year.

Keywords: Bluetooth Low Energy (BLE); Wi-Fi; proximity sensor; android things; cloud messaging; shopping list

1. Introduction

The shopper marketing agency V360 [1] found that most people get distracted and forget what they went out for shopping, and 92% of people end up coming home without the item they intended to buy. Half of those surveyed say that they forget on a regular basis [2]. The research published in [3] suggests that consumers should always use shopping lists as they cannot predict when they are likely to forget. However, making a shopping list for each grocery item requires to check the remaining quantity of that item from different containers and canisters, and then write down the items that are finishing soon on a paper or in a smartphone. This manual process can be time consuming and tiresome.

In this paper, a novel Internet of Things (IoT)-connected smart canister system is developed, which automatically senses the quantity of the item in the canister, sends the data to a central hub using Bluetooth Low Energy (BLE), and the hub sends a cloud message to the consumer's smartphone app using the Internet. The hub and the smartphone app display the item quantities in the canisters and add the items that are about to finish to a digital shopping list. The automatic generation of the shopping list removes the burden of manually checking each item before going to the shops and gives peace of mind to the consumers. The shopping list can be accessed from any place as long as the smartphone is connected to the Internet.

In the proposed method, each canister or bottle cap contains a low-power and small-size electronic device comprising a proximity sensor, a system-on-chip microcontroller with BLE, detector switch, and a rechargeable battery. The device wakes up from low power sleep mode whenever it is put on the canister, then it measures the item quantity and sends the data to a central hub using BLE. The central hub is wirelessly connected with the home Wi-Fi and sends a cloud message using the Internet. The smartphone app is notified immediately using push notification, then it downloads the item quantity and battery level data from the cloud, and then adds the items to the shopping list that are less than a threshold level.

Using the menu of the hub app, a new canister can be added, its configuration can be edited, or existing canisters can be deleted. A prototype of the proposed system with three canister devices, one hub, and the smartphone app is developed and tested successfully.

Smart homes and IoT are automating the ability to control and monitor items around the house—from window shades to pet feeders—with a smartphone or using voice command. Some activities, like setting up a lamp to turn on and off [4], monitoring a baby's diaper wetness [5,6], controlling and monitoring chicken eggs in incubator [7], are some examples. Several works are found in the literature on generating shopping lists using IoT and smart home technology. In [8], a smart refrigerator is designed with three cameras to see inside from anywhere using a smartphone. In this technology, the user first sees the image of the items in the refrigerator and then manually makes a shopping list of the items that are finishing soon. Thus, this process of generating a shopping list is not automated. Moreover, some items may not be visible by the camera if the refrigerator is crowded with many food items. In our proposed system, the bottles can be scattered in any place around a 100-meter radius of the central hub and do not need to be under direct line of sight of the camera.

In [9–11], smart water bottles are designed that track every sip by a sensor and send total water consumption to a smartphone app using Bluetooth. The bottle in [9] also glows to give a visual reminder, and the bottle in [10] generates vibration to drink water to achieve hydration and fitness goals. In [11], water level data are sent to a smartphone using a Global System for Mobile Communications (GSM) modem. However, the current consumption of a GSM modem in normal operation is 250 mA and can rise up to 1 A while transmission [12], which is not practical for a battery-operated IoT device. The Smart Container in [13] senses when coffee pods are running low and automatically reorders the item from Amazon. This device connects with home Wi-Fi and has a battery life of around one year. This product is not general for all kinds of items, rather it is designed for coffee pods only.

In [14], a smart canister system is proposed where the remaining item is detected using a proximity sensor. It is linked to a smartphone via a Wi-Fi, and the data received are stored in a cloud. The Android app (which is under development) is supposed to notify when items are running out and automatically add such items to a shopping list. As this canister connects with the smartphone directly using Wi-Fi, it will consume 10 times more power than BLE, even if they are performing the same tasks [15]. This will significantly reduce the battery life, and it will need frequent recharging, which is an overhead. In this paper, the proposed canister or bottle cap contains an ultra-low power and small-size electronic device with a rechargeable battery. As it uses BLE to communicate with the central hub, the power consumption of the device is very low, and it can have a long battery life.

2. Materials and Methods

Whenever the cap is put on the canister, the device wakes up from low power sleep mode, measures the item quantity using a proximity sensor, and sends the data to a central hub using BLE. The central hub—that can be attached on a kitchen refrigerator—contains an LCD with a touchscreen to configure the canisters and to display the item quantities, connection status, and battery levels of each canister. The central hub 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; it downloads the data from the cloud and adds the items to the shopping list that are running below a threshold level. The overall system architecture of the proposed smart canister system is shown in Figure 1.

The different components of the system are briefly described below.

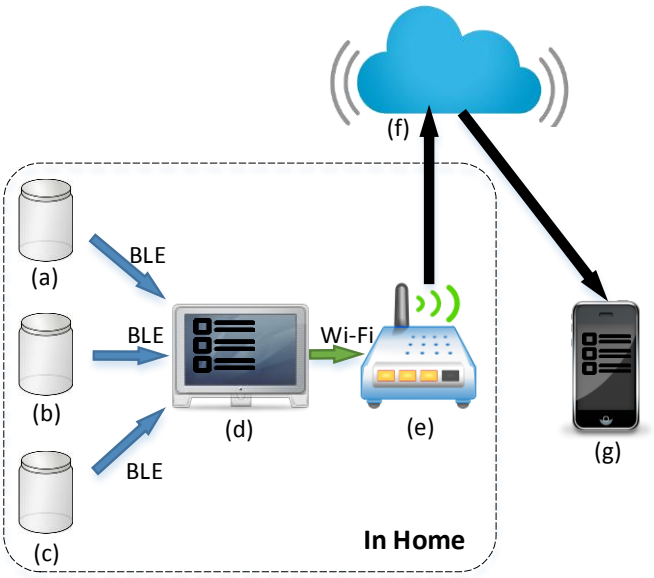


Figure 1. The architecture of the smart canister system. (**a**–**c**) are smart canisters with the electronic device placed in the caps. The canisters are connected wirelessly with the central hub (**d**) using BLE. The central hub displays the item quantities, connection status, and battery levels of each canister. The hub is connected with the home router (**e**) using Wi-Fi. The hub sends cloud message (**f**) using the Internet to the smartphone app (**g**) to sync the data between hub and smartphone. The items that are running below a threshold level are automatically added in the shopping list of (**d**,**g**).

2.1. Smart Canister

The smart canister measures the remaining quantity of the item inside it and sends the data to the central hub whenever the cap is put on the canister. A hardware device is designed for this purpose—attached at the bottom of the canister cap. The key design challenge of the device is lowering the power consumption as it is powered by a battery. The hardware and firmware of the device are briefly described below.

2.1.1. Hardware

The block diagram of the hardware unit of the device inside the cap is shown in Figure 2.

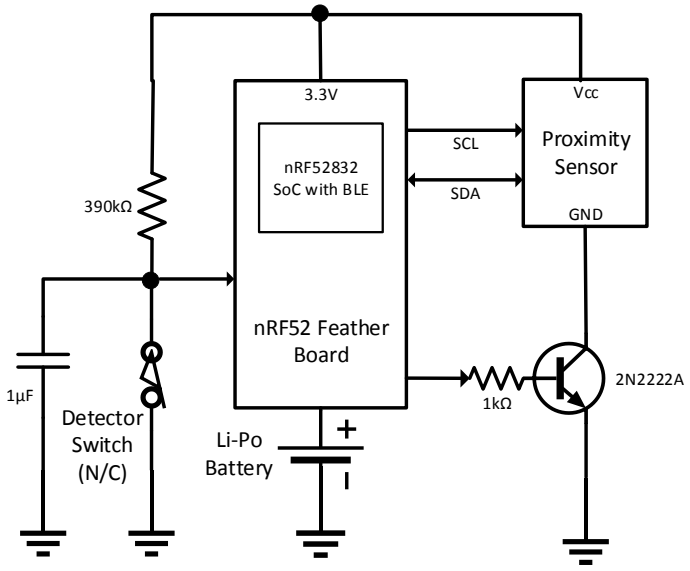


Figure 2. Block diagram of the hardware device that is attached at the bottom of the canister cap.

A low-power, small-size system-on-chip (SoC) micro-controller with BLE, nRF52832 [16], is used as the processing and wireless communication unit. The micro-controller contains an industry-standard ARM Cortex M4F MCU, 512 kB in-system programmable flash memory, 64 kB RAM, 2.4 GHz BLE transceiver, 8 analog-to-digital converter (ADC) channels, general-purpose input/output (GPIO), timer, I2C, and many other peripherals. It has low-power sleep modes, and it is suitable for systems where ultra-low power consumption is required to increase battery life.

In this project, the nRF52 Feather [17] board is used, which consists of the nRF52832 microcontroller, USB-serial converter for efficient programming and debugging, a connector for 3.7V 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 do not 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 3.7 V Li-Po rechargeable battery [18] with a capacity of 2500 mAh is used as the power source for this hardware unit. In order to monitor the battery voltage—to know when recharging is needed—an ADC pin is connected to the middle point of a voltage-divider circuit that is powered by the Li-Po battery.

A proximity sensor [19] is used to measure the distance from the cap to the upper surface of the item inside the canister. Thus, this sensor measures the height of the empty portion in the canister. By subtracting the empty portion height from the maximum possible height in the canister, the filled-up portion in the canister is calculated. The sensor [19] uses a precise clock to measure the time it takes for infrared light to bounce back from the surface, independent of the surface reflectance. It can measure distance up to 25 cm. The sensor is connected with the microcontroller using the I2C interface. In order to reduce power consumption, the sensor is cut off from power when it is not used. An NPN transistor [20] is used to connect and disconnect the sensor's ground pin from the power supply ground. The base of the transistor is controlled by a GPIO pin of the microcontroller, as shown in Figure 2. The base current flows only for a few seconds during stabilization time and distance measurement, and then the current is made zero by making the GPIO pin of the microcontroller to LOW to cut off the power of the distance sensor. Thus, the base current has no significant effect on battery life.

The normally closed (N/C) and common (COM) terminals of a snap action detector switch [21] are used to detect whether the cap is put on the canister. The lever of the switch gets pressed by the top wall of the canister when the cap is put on, and the switch's N/C contacts get opened; the lever of the switch gets un-pressed when the cap is removed from the canister and the switch's N/C contacts get closed. The switch is connected to a GPIO interrupt pin of the microcontroller, and the interrupt was configured to be triggered on the rising edge. A 1 µF capacitor is connected from the MCU pin to the ground to filter out the bounce signal from the switch. An external pull-up resistor of 390 kΩ—as shown in Figure 2—is used to make the pin high when the switch is pressed. The nRF52832 has internal pull-up resistors (R_{pu}) with a value of approximately 13 kΩ [16]. If the internal pull-up resistors are enabled, this branch through the switch will continuously consume 3.3 ÷ 13k = 0.25 mA current as long as the cap is removed from the canister. This is a significant current. To solve this problem, the internal pull-up resistor was disabled and a higher value external pull-up resistor of 390 kΩ is used. In this case, the branch consumes only 3.3 ÷ 390k = 8.46 µA current (measured 8.24 µA in real-time) when the cap is removed from the canister. Experiments showed that if a higher value of external resistor, such as 1 MΩ, is used, then too much voltage drops, and the microcontroller pin does not recognize a rising event interrupt. Thus, 390 kΩ external pull-up resistor is a good design choice. Now, when the bottle cap is put on the canister, the detector switch gets pressed and the switch's N/C contacts get opened. This causes the microcontroller pin to go high and it triggers a rising interrupt event. As the input resistance of the microcontroller pin is high, the sink current is approximately zero (measured 0.01 µA in real-time).

2.1.2. Firmware

In this proposed system, the canister devices send data to the central hub using BLE. The canister devices are configured as *peripheral* and the hub is configured as *central* [22]. A high-performance

Bluetooth 5 qualified protocol stack for the nRF52832 SoC—referred to as *SoftDevice*—is implemented that contains complete stack with Generic Access Profile (GAP), Generic Attribute Profile (GATT), Link and other layers [23]. A custom *service* with a Universally Unique ID (UUID) of 00001400-0000-1000-8000-00805f9b34fb is created in the GATT profile. Under this service, a custom *characteristic* with a UUID of 00001401-0000-1000-8000-00805f9b34fb is created. The characteristic has a data length of 2 bytes—one byte for proximity sensor data and another byte for battery level. Its *notify* property is set so that the central hub gets a notification whenever the data in this characteristic is updated. The device advertises with its GAP data so that it can be found by the central hub. Once the central hub scans and finds the device, a connection is established.

A real-time operating system (RTOS), known as *FreeRTOS* [24], is used in the firmware. The firmware is designed by using event response—known as *call-backs*—without always running codes inside a loop. The firmware goes to low power sleep mode and waits for an event to happen to wake up. Whenever the cap is put on the canister, the detector switch gets pressed, the *cap_closed* interrupt is triggered, and the microcontroller wakes up. The interrupt service routine (ISR) then starts a timer and also applies power to the proximity sensor by making the GPIO pin HIGH that is connected with the base of the transistor. Within the delay from the timer, the canister 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 I2C and ADC peripherals are enabled, proximity sensor data and battery levels are read, then the I2C and ADC peripherals are disabled. The sensor and battery level data are then written on the BLE *characteristics* and it causes a call-back function to be called in the central hub for reading the data. Finally, the power is cut off from the sensor by making the GPIO pin—that controls the transistor—LOW. Then the program goes to low power mode and waits for another event to happen. The actions executed by the device after the cap is put on the canister are shown in Figure 3.

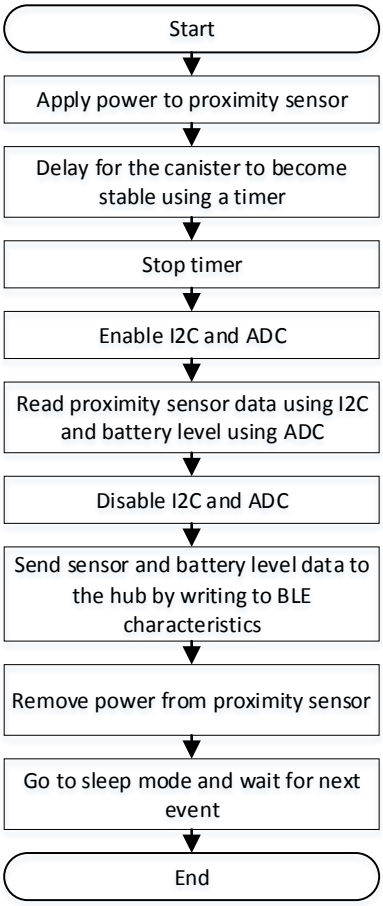


Figure 3. Flowchart of actions after the cap is put on the canister.

To make the program power efficient, the following configurations were done:

- The frequency of the RTOS tick interrupt (referred to as configTICK_RATE_HZ) was reduced from 1024 Hz to 4 Hz to reduce power. This change caused 200 µA current reduction. The tick interrupt is used to measure time. Therefore, a higher tick frequency means the timer can measure time to a higher resolution. The RTOS scheduler will share processor time between tasks of the same priority by switching between the tasks during each RTOS tick. A high tick rate frequency will therefore also have the effect of reducing the "time slice" given to each task [24]. The system clock of the microcontroller (referred to as SystemCoreClock) is 64 MHz and it is not changed. Thus, the microcontroller executes instructions at a high speed without sacrificing the performance.
- No floating point numbers were used because the floating point unit (FPU) consumes significant power. To take the microcontroller in low power mode, the exception flags and the pending FPU interrupts were cleared [25].
- The DC/DC converter of the microcontroller is enabled instead of low dropout (LDO) regulator [26].
- The advertising BLE connection interval was set to 20 ms, and the transmission power level was set to 0 dB to balance between power consumption and performance. We did an experiment by reducing the transmission power level to −30 dB, however, no significant reduction of current consumption was noticed, but created problem during connection.

2.2. Central Hub

In the proposed system, the central hub receives data from multiple smart canisters using BLE and sends a cloud message using the home Wi-Fi to the smartphone app. The hub is designed with the state-of-the-art touch-screen-based LCD unit that can be attached to a refrigerator or on the kitchen wall, showing item quantities, connection status, and battery levels of each canister. The hardware and app parts of the hub are described below.

2.2.1. Hardware

A Raspberry Pi (RPi) v3 [27] single board computer is used as the main processing unit. It contains a 1.2 GHz 64-bit quad-core ARMv8 microprocessor, 1 GB of RAM, micro SD card slot supporting up to 32 GB, LCD interface (DSI), on-board BLE and Wi-Fi module, and other built-in hardware peripherals. A 7-inch capacitive touch LCD [28] is interfaced with the RPi using the DSI and the I2C port. The LCD has 24-bit color depth and the screen resolution of 800 × 480 pixels. The power supplies for the RPi board and the touch LCD are supplied using 110 V AC to 5.1 V DC adapters [29].

2.2.2. App

The Android Things [30] embedded operating system is installed on a 16 GB secure digital (SD) card of the RPi board. Android Things is recently developed by Google, aimed to be used with low-power and memory constrained IoT devices. The platform can run any program targeting Android and it also comes with other libraries for accessing hardware peripherals, such as BLE, Wi-Fi, and I2C of the RPi. The device gets connected to the home router using Wi-Fi and can access the Internet.

The first screen of the app contains a list-view box. Each record in the list-view box shows the name of the item in the canister, percentage of the item quantity, connection status, and the battery level. The list-view box reads this information from a comma-separated values (CSV) file. This file contains the current properties of each canister, such as item name, maximum height, threshold height, current height, BLE MAC address, battery level, and whether connected with BLE or not.

The app contains a settings menu for configuring the smart canisters. From this menu a canister can be added, edited, or deleted. To add a canister, the user manually enters the item name of the canister (such as bean or coffee), maximum possible item height, and threshold height. The item will be automatically added in the shopping list when the current height of the item goes below the threshold height. Then 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 smart canister's MAC address from the list. The item name, maximum height, threshold height, and MAC address are saved in the CSV file. A canister can be deleted by making a long click on item name of the canister.

After one or more canisters are added, the user presses the "connect" button to get connected with all the canisters using BLE. The app scans the nearby BLE devices and sends a connection request to those devices whose MAC addresses match the MAC addresses stored in the CSV file. As there are multiple canisters, multiple BLE connections are created using an array of instances of BLE manager objects.

Now, whenever the cap of the canister is put on, the hardware device in the cap measures the height from the cap to the item surface and measures the battery level and sends the data to the hub. The hub app is notified with new data, it reads the 2 bytes from the BLE *characteristic*, and it stores the data in the CSV file. Whenever the CSV file is updated, it refreshes the list-view in the first screen showing the item quantities and battery levels and also sends a cloud message—with the content of CSV file—to the smartphone app for data synchronization. Sending the cloud message is done with a Hypertext Transfer Protocol (HTTP) request. The message is sent using a *Server key* that is generated from the cloud server where the smartphone app is registered. The steps executed by the hub whenever BLE data are received are shown in Figure 4.

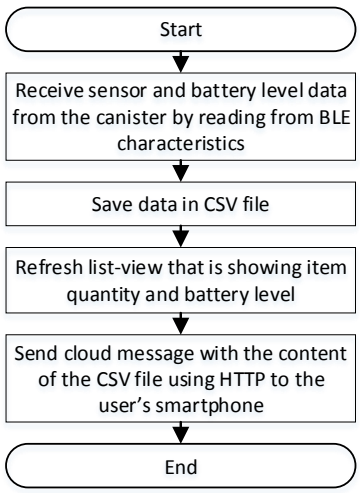


Figure 4. Flowchart of actions when BLE data are received.

2.3. Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) [31] is used to send data from the hub to the customer's smartphone app. FCM is a cross-platform messaging solution that reliably delivers messages at no cost. Using FCM, a smartphone app can be immediately notified whenever new data are available to sync. A message can transfer a payload of up to 4 KB to a client app.

2.4. Smartphone App

The smartphone app is developed for the Android platform. It is written with a similar structure of the hub app with some additional features. The first screen of the app contains a list-view box showing the name of the item in the canister, the percentage of the item quantity, connection status, and the battery level. It also contains a button "Shopping List" that is used to display only the items whose current height is below the threshold height.

The app is added and registered in the FCM for receiving the push notification. A service [32], 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 message in the local smartphone in a CSV file. The list-view box in the first screen and shopping list is updated with the CSV file whenever a new message arrives. In this way, the data are synchronized between the hub in

the home and the user's smartphone at whatever place the user may be in. The flowchart in Figure 5 shows the actions taken by the smartphone whenever cloud message is received.

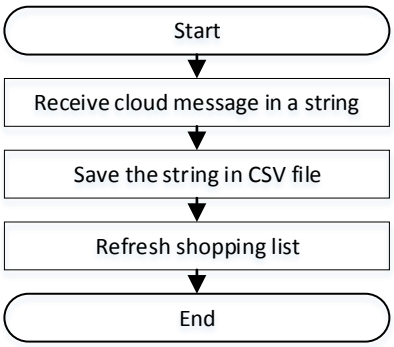


Figure 5. Flowchart of actions when the cloud message is received.

3. Results

A prototype of the proposed smart canister system comprising three canisters, one central hub, and a smartphone app has been developed, calibrated, and tested successfully. A photograph of the device, that is attached under the canister cap, is shown in Figure 6. The cap has a diameter of 9 cm. Two devices were attached with two caps and they are used to measure the item quantity of rice and lentils as shown in Figure 7.

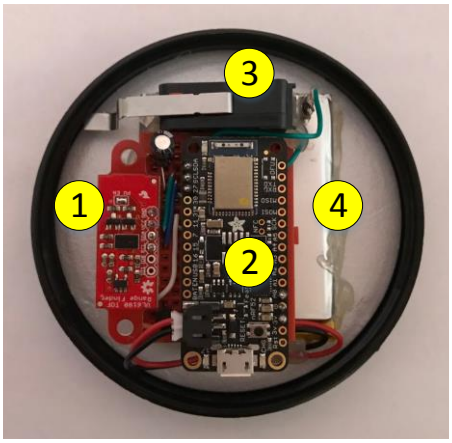


Figure 6. Photograph of the device attached under the canister cap—(**1**) proximity sensor, (**2**) SoC microcontroller with BLE and battery charger, (**3**) detector switch, and (**4**) Li-Po rechargeable battery.

Figure 7. Smart canisters containing rice (left) and lentils (right).

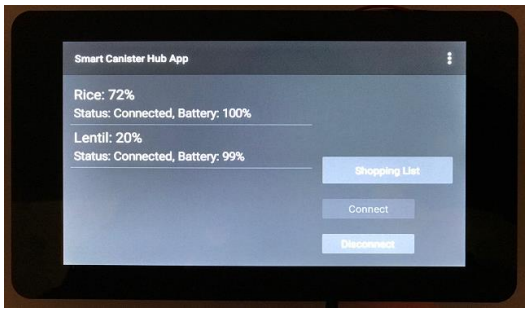
The third device prototype was developed on a breadboard, and an ammeter was connected through the positive wire of the battery to measure the current consumption. The measured current consumptions of the device at different states are shown in Table 1.

Table 1. Measured current consumption of the device at different states.

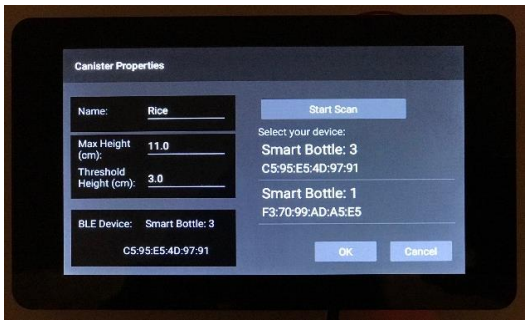
Device State	Current Consumption (μA)
Advertising	~600
Connected and cap not on the canister	233
Connected and cap on the canister	224

The BLE of the device will advertise only when it needs to be discovered by the central hub. Once discovered and connected, the device consumes 233 μA when the detector switch is un-pressed—i.e., the cap is not put on the canister, and the device consumes 224 μA when the detector switch is pressed—i.e., the cap is put on the canister. When the cap is put on, the device wakes up from sleep mode, measures and transmits data for a short amount of time, and again goes to sleep mode. The prototype uses a Li-Po rechargeable battery with a capacity of 2500 mAh. As the device will be connected and the cap will be on the canister for most of the time, the battery life of the device is approximately 2500 ÷ 0.224 = 11,160.71 hours, or 465 days.

The photographs of the central hub are shown in Figure 8. The hub's display shows the item quantities, connection status, and battery levels of each canister as shown in Figure 8a. A canister can be added, edited, or deleted from the hub app. Figure 8b shows the properties of the canister that contains rice. The hardware of the hub is designed with Raspberry Pi board interfaced with LCD is shown in Figure 8c.



(**a**)

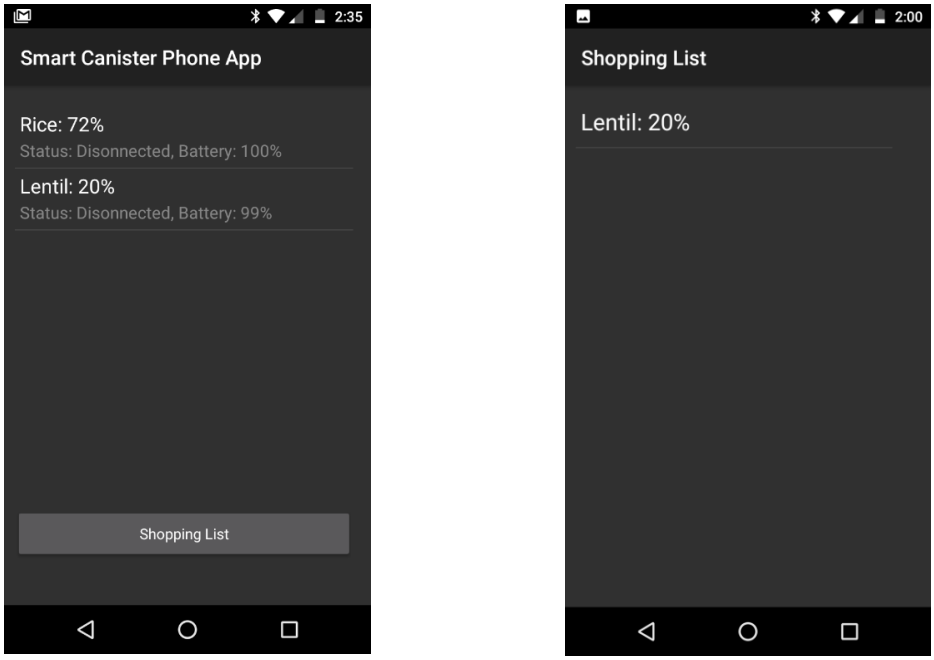


(**b**)

(**c**)

Figure 8. Photograph of the central hub—(**a**) front screen showing item names in the canisters, connection status, and battery levels; (**b**) Configuration of canister properties; (**c**) bottom view—Raspberry Pi board interfaced with LCD.

The screenshots of the smartphone app are shown in Figure 9. The app shows the item quantities, connection status, and battery levels of each canister in the front screen, as shown in Figure 9a. When the "Shopping List" button is pressed, the app shows the list of items that are below the threshold level, as shown in Figure 9b.



(a) (b)

Figure 9. Screenshot of the smartphone app—(**a**) front screen showing item names in the canisters, connection status, and battery levels; (**b**) shopping list showing the items that are below the threshold levels.

The smart canister system is tested in the real world by taking the smartphone several miles away from the canisters and the hub. The phone was out of the range of home Wi-Fi and was connected with the Internet using the cellular network. When the cap was put on the canister, the item quantity, connection status, and battery level got updated immediately in the smartphone app.

4. Discussion

After the IoT device was attached at the bottom of the cap and the cap was put on the canister, it was found that the proximity sensor was not accurately measuring the distance from the cap to the item surface. To solve this problem, the sensor data were calibrated. Several data points were gathered by recording the true distance and the measured distance by the sensor. Then a plot was made, and Equation (1) was derived using the curve fitting technique. In Equation (1), d_t is the true distance in millimeter (mm) and d_m is the measured distance by the sensor in mm. The hub app and the smartphone app implement Equation (1) to find the true distance from the sensor data.

$$d_t = 0.9393 d_m - 9.7424 \quad (1)$$

The coefficient of determination, denoted by R^2, for the linear Equation (1) is 0.997 or 99.7%. R^2 is a statistical measure of how close the data are to the fitted regression line, and it ranges from 0% to 100%. In general, the higher the R^2, the better the model fits the data, so the accuracy of the calibrated sensor data is more than 99%.

When canisters of different sizes and shapes are used, the height may not properly signal the item quantity as their length and width will vary. One way to solve this is to take the length and width of

the canister (or diameter for cylindrical canister) during canister configuration, as shown in the screen in Figure 8b, and calculate the volume from the measured height. The volume, instead of height, will give more accurate information when canisters of different shapes and sizes are used.

The hardware device prototype under the canister cap is built using microcontroller and proximity sensor break-out boards. Break-out boards are used to reduce prototype development time and soldering complexity. However, if bare chips are used in a custom designed printed circuit board (PCB), the device can be made by spending less than 30 USD including the battery. The price will be reduced more when it is produced in bulk quantity. The central hub can be developed with 100 USD. The FCM has no cost, and the smartphone app can be downloaded freely.

The future work includes making PCB with surface mount device (SMD) components to reduce the device size and cost, using MOSFET [33] to cut-off power from the sensor, implementing the system for multiple users, and developing the smartphone app in the iOS platform.

5. Conclusions

In this paper, a novel smart canister system is developed that automatically measures the item quantities and generates a digital shopping list in the smartphone that can be accessed from any place using the Internet. The canister cap contains a low power IoT device with a battery life of more than a year.

Author Contributions: Conceptualization, methodology, software, validation, analysis, investigation, resources, writing—original draft preparation, writing—review and editing, visualization, supervision, project administration, funding acquisition—by T.K.

Funding: This research was funded by James H. Brickley Endowment for Faculty Award of Eastern Michigan University.

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

References

1. V360 Shopper Marketing Agency. Available online: http://v360.ie/ (accessed on 7 May 2019).
2. Weston, C. Most People Forget to Buy What They Went to Shop for. Available online: https://www.independent.ie/business/personal-finance/most-people-forget-to-buy-what-they-went-to-shop-for-36857147.html (accessed on 7 May 2018).
3. Fernandes, D.; Puntoni, S.; Osselaer, S.; Cowley, E. When and Why We Forget to Buy. *J. Consum. Psychol.* **2015**, *26*, 363–380. [CrossRef]
4. Khan, T.H. A Wi-Fi based architecture of a smart home controlled by smartphone and wall display IoT device. In Proceedings of the 1st International Virtual Conference on Multidisciplinary Research (IVCMR 2018), Walnut, CA, USA, 14 November 2018; Volume 3, pp. 180–184.
5. Khan, T.H. A noninvasive smart wearable for diaper moister quantification and notification. *Int. J. Electr. Comput. Eng. (IJECE)* **2019**, *9*, 2848–2862.
6. Khan, T.H. A smart wearable gadget for noninvasive detection and notification of diaper moister. In Proceedings of the IEEE International Conference on Electro information Technology (EIT 2018), Rochester, MI, USA, 3–5 May 2018; pp. 240–244.
7. Adnyana, G.; Piarsa, N.; Wibawa, K. Internet of Things: Control and Monitoring System of Chicken Eggs Incubator Using Raspberry Pi. *Int. J. Internet Things* **2018**, *7*, 16–21.
8. Samsung Refrigerator with Family Hub. Available online: https://www.samsung.com/us/home-appliances/refrigerators/3-door-french-door/26-cu-ft--capacity-3--door-french-door-refrigerator-with-family-hub--2-0-rf265beaesr-aa/ (accessed on 7 May 2019).
9. Hidrate Spark Smart Water Bottle. Available online: https://hidratespark.com/ (accessed on 7 May 2019).
10. Ozmo Smart Bottle. Available online: https://www.ozmo.io/ozmo-smart-bottle/ (accessed on 7 May 2019).
11. Pankajavalli, P.; Saikumar, R.; Maheswaran, R. Hydration reminding smart bottle: IoT experimentation. In Proceedings of the IEEE Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, 21–22 April 2017; pp. 1–5.
12. Pandya, V.; Shukla, D. GSM modem based data acquisition system. *Int. J. Comput. Eng. Res.* **2012**, 2, 1662–1667.

13. WePlenish Java Smart Container. Available online: https://weplenish.com/products/weplenish-java-smart-container/ (accessed on 7 May 2019).
14. Singh, A.; Ragav, S.; Sreenivasan, V.; Murugan, S. Home Automation: Smart Canister using Internet of Things. *Int. J. Sci. Res. (IJSR)* **2018**, 1461–1463. [CrossRef]
15. Prasad, P. BLE vs. Wi-Fi: Which is Better for IoT Product Development? Available online: https://www.cabotsolutions.com/2018/02/ble-vs-wi-fi-which-is-better-for-iot-product-development (accessed on 7 May 2019).
16. nRF52832 SoC. Available online: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52832 (accessed on 7 May 2019).
17. Adafruit Feather nRF52 Bluefruit LE-nRF52832. Available online: https://www.adafruit.com/product/3406 (accessed on 7 May 2019).
18. Lithium Ion Polymer Battery-3.7v 2500mAh. Available online: https://www.adafruit.com/product/328 (accessed on 7 May 2019).
19. SparkFun ToF Range Finder Sensor-VL6180. Available online: https://www.sparkfun.com/products/12785 (accessed on 7 May 2019).
20. NPN Bipolar Transistors (PN2222). Available online: https://www.adafruit.com/product/756 (accessed on 7 May 2019).
21. Snap Switch SPDT Chassis Mount. Available online: https://www.digikey.com/product-detail/en/e-switch/LS0851503F040C1A/EG4527-ND/1628264 (accessed on 7 May 2019).
22. Davidson, R.; Townsend, K.; Wang, C.; Cufí, C. *Getting Started with Bluetooth Low Energy Tools and Techniques for Low-Power Networking*; O'Reilly Media Inc.: Sebastopol, CA, USA, 2014.
23. S132 SoftDevice. Available online: https://www.nordicsemi.com/Software-and-Tools/Software/S132 (accessed on 7 May 2019).
24. Free RTOS Customization. Available online: https://www.freertos.org/a00110.html (accessed on 7 May 2019).
25. 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 7 May 2019).
26. Optimizing Power on nRF52 Designs. Available online: https://devzone.nordicsemi.com/nordic/b/blog/posts/optimizing-power-on-nrf52-designs (accessed on 7 May 2019).
27. Raspberry Pi. Available online: https://www.raspberrypi.org (accessed on 7 May 2019).
28. Raspberry Pi LCD-7" Touchscreen. Available online: https://www.sparkfun.com/products/13733 (accessed on 7 May 2019).
29. DC Power Supply. Available online: https://www.sparkfun.com/products/13831 (accessed on 7 May 2019).
30. Android Things. Available online: https://developer.android.com/things/get-started/index.html (accessed on 7 May 2019).
31. Firebase Cloud Messaging. Available online: https://firebase.google.com/docs/cloud-messaging (accessed on 7 May 2019).
32. Android Service. Available online: https://developer.android.com/guide/components/services.html (accessed on 7 May 2019).
33. N-Channel MOSFET 30V 24A. Available online: https://www.digikey.com/product-detail/en/infineon-technologies/IRL2703PBF/IRL2703PBF-ND/811700 (accessed on 7 May 2019).