Bluetooth home alert system

COMP3210 Networking Exemplar coursework report

Sarunas Iljeitis
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
si1g16@soton.ac.uk

***Abstract*—The report describes a home alert system prototype consisting of a set of independent low-energy devices, system hub and a cloud service. The system employs Bluetooth Low Energy technology for communication with devices and is compatible with both IPv4 and IPv6 networks. The user is able to manipulate the system via a control hub graphical interface, view statistics on the cloud platform and administrate system subscribers who receive notifications about the system events.**

***Index Terms*—BLE, IoT, Home security, Smart home**

I. Introduction

Every year there is an significant increase in house burglaries during the warm summer months. According to Lloyds banking group press release unforced burglary claims had increased by more than half between March 2018 and April 2018 [1]. The cause for this is clear - people tend to leave their windows and doors unlocked or opened during the hot period. Students tend to fall victims even more often as more than 660 000 students are burgled every year in UK alone [2]. As landlords are usually not concerned about their tenants safety, the majority of one and two storeys houses which are regularly rented by forgetful students, become easy targets to night burglars. It is only natural to be tempted of leaving an opened window in ones bedroom mid-August as temperatures even at night do not fall below 20 degrees Celsius.

Even simple stand-alone house security devices decrease burglary possibility around three times study finds [3]. Therefore, even a low-cost Do-It-Yourself (DIY) type alert device would considerably increase household security at least against simple burglaries. House alarm system installations are usually complicated, inflexible and troublesome when proper wiring is needed. Therefore, in today's world, low-power consumption and wireless data transfer are almost mandatory requirements for any smart home system. Common users also demand the ability to control their home systems via smart tablets and smartphones, as well as access event logs via interactive graphical user interfaces. An uncomplicated, extendable and easy to setup and use alarm system would be a favourable low-effort measure against residential household burglaries.

We propose an IoT home alert system prototype as a low-price product alternative to professional home alarm systems. It is not aimed to replace full-size home security systems but provide a way for house tenants to be notified of possible intruders or trespassers utilizing well-known consumer technologies. The system consists of multiple independent Sensor and Ringer units which are connected to main house system hub via Bluetooth Low Energy technology. The system Sensor units employ an ultrasonic sensor to monitor movement over a certain narrow area e.g. a window or a corridor. System Ringer units are capable of receiving messages from the hub and are used to create a loud sound alarm. Depending on the setup, the system hub may forward events to a cloud instance where they are logged and visualized. The system owner may also configure the cloud service to send out email notifications to a list of interested subscribers. The system is controlled, configured and may be expanded via the hub administrator control panel which provides an overview of the local deployment.

A possible alert system use case could be as a night time window alarm device employed during the summer months than the owner is asleep but still tends to leave house windows opened. A loud signal would wake up the tenants as well as scare away intruders. Due to the systems flexible nature, it could also be used to monitor movement around the property and even non-intrusively track elderly family members movement inside the house.

II. Background research

This section explains key concepts required in order to understand proposed alert systems advantages and usefulness.

A. Wireless distributed system

In order to create an efficient, robust and convenient alarm system the raw sensor data has to be processed, interpreted and acted upon on different logical levels. A combination of inexpensive dedicated sensor units together with a system control module and an off-site storage and analysis platform provides the best overall system result utilizing each segments strong points.

It is a classic case of a distributed IoT system where data gathering, node management, external services and user interactions are allocated to separate system components [4]. It is common to use power efficient, compact, independent agents for specific tasks. A wired or wireless link may be used to transfer modulated data to a regional control unit depending on particular deployment trade-offs. The regional

system hub is the next infrastructure level which interprets sensor signals, alter sensors setup accordingly and forwards significant information to the higher-level platform. The cloud, theoretically, has no processing, storage or power usage limitations, therefore is capable of handling any amount of gathered data on user request and forward it to other services. It is of highest demand for the cloud platform to be compatible with any regional hub client, therefore, a reliable, universal cloud interface is required for it to function.

B. Wireless communication

As alarm system wiring can be a hassle and as the data throughput from sensors is negligent, a wireless communication between individual units and the hub could prove advantageous. Common wireless communication protocols used in the IoT networks are grouped by their data transfer rates, power consumption and range. As the system hub and sensor units communication is expected to be via short or medium distances, several alternative communication protocols can be used.

ZigBee is a popular low-energy communication protocol for IoT systems. ZigBee supports a range of setups and can be used in very small or large IoT networks. It is largely used in industrial level technologies and not that common in consumer goods [5]. Bluetooth Low-Energy (BLE) protocol is very similar to ZigBee as it is a stand-alone protocol (not compatible with regular Bluetooth devices), it uses the same 2.4GHz frequency as ZigBee but is widely used in consumer devices. Some studies even suggests BLE being more energy efficient than it's counterpart [6]. The 6LowPAN protocol is the most widely used across IoT networks as it supports direct node communication and mesh networks [7]. It has a low data throughput, low power consumption and can be configured to work on several frequencies. For the suggested prototype application non of the low power protocols have a distinct advantage over the others. BLE protocol has been widely adapted by DIY electronics project community members, it's components and documentation is easily available, therefore, it is a suitable choice for such project implementation.

C. Bluetooth Low-Energy (BLE)

The following BLE protocol overview follows the "Getting Started with Bluetooth Low Energy" book [8]. Bluetooth Low-Energy is a one to many connection type protocol. Meaning that individual system nodes can either be Central or Peripheral nodes but not both. Peripheral devices can broadcast advertisement packets announcing their presence. Central devices listen for advertisement packets and can request additional information about the peripheral node. Small amounts of data can be sent by BLE devices to all surrounding nodes using the broadcast mode but a more reliable way is to establish a connection between two devices first. A Central device upon receiving a suitable advertisement packet from a nearby peripheral node can request a connection/pairing. Once the link is established Master (central) device is responsible for maintaining the connection and data can be transferred both ways.

The BLE network stack consists of several layers.

The Physical layer, apart from obvious analogue-digital conversion, is responsible for frequency hoping between 37 data channels when the connection is established and 3 peripheral device advertisement channels. This is used to accomodate numerous BLE enabled devices in the same area with little interference.

The Link layer defines the BLE device role/mode such as Peripheral - Central (Slave - Master). It is also responsible for establishing and maintaining connection. BLE devices use 48-bit unique identifiers similar to MAC addresses which together with encryption are used in the Link layer.

The lower levels of the protocol are abstracted from the host device using the Host Controller Interface. All the upper layer data is encapsulated in two types of packets: ATT - for data transfer and SMP - for pairing and encryption messages. The L2CAP logical layer which is common to all Bluetooth protocols (Classic, Low-Energy, "Smart") governs these packets transmission.

The mentioned Attribute layer (ATT) is the BLE way of structuring data moving between Master and Slave nodes. The Slave node can also be referred to as the Server and Master not as the Client. To transfer data the client can request read or write events of a particular server attribute which are uniquely identified using a 16-bit UUIDs.

The final two layers are Generic Attribute Profile (GATT) and Generic Access Profile (GAP). GATT structures device attributes into pre-defined Services. An individual service could respond to a particular device role like a heart monitor. Every service is then divided into Characteristics which contain a single peace of data (e.g. heart rate, battery voltage, pressure). The GAP profile is the device configuration layer. It is used to define user role, mode, encryption and connection.

D. IPv4 mapped to IPv6

For the system to smoothly function via both versions of the Internet Protocol (IP), the application implementation needs to employ IPv4 to IPv6 address mapping [9]. The application is configured to expect IPv6 traffic and if IPv4 traffic is received, it seamlessly prepends a special IPv6 prefix to it, thus solving any compatibility issues. Applications configured this way will work regardless of the network configuration they are deployed in.

III. Design and Implementation

A. Overall system design

The proposed alert system prototype is a distributed IoT system consisting of two types of low-level units (Sensors and Ringers), a system hub, a cloud platform and an external mail service. The complete system view is displayed in Figure 1.

The system hub (regional hub) is a Raspberry Pi microcontroller representing the core of the system. Two separate processes run on the system hub at all times: BLE event handler and a web application. BLE handler maintains connection

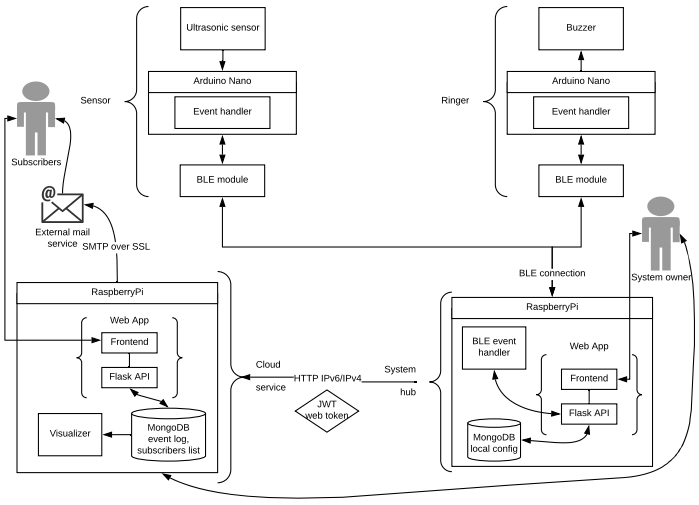

Fig. 1. Complete system view

with the Sensor and Ringer units, interprets their messages and issues commands (e.g. raise alarm, reset, re-measure). A Flask web server provides a systems administrative interface for the user which is used to add and remove system hub devices, configure the hub and respond to events. The system hub data (e.g. list of belonging Ringers) is stored in a non-relational MongoDB database on the Pi.

The Sensor and Ringer units implementation is explained in more detail in the following section. In essence, both unit types are implemented using Arduino Nano microcontrollers to minimize power consumption, which is also connected to either an ultrasonic sensor or a Piezo speaker via a digital data wire and uses a cost-efficient HM-10 CC2541 BLE serial module for communication with the hub. Upon establishing connection, units can either send or receive commands from the hub by writing into BLE Serial interface and the HM-10 module will seamless convert Serial message strings to BLE packages and send them out to the destination.

The alert system events are stored off-site on the cloud platform. Individual system hubs connect to the cloud using the JSON Web Tokens (JWT) standard. Every registered system hub will report it's events to the cloud where they will be stored in a MongoDB database. The cloud platform also provides a web application for user interaction. It displays system event log, visualizes it in a graph and is used to modify subscribers email list. When an alarm goes off, the cloud application uses an external web service to send out warning emails to all of it's system subscribers, that way providing an alternative means of notifying users of an alarm event.

B. Units and system hub implementation

As the report author was responsible for the BLE communication, the Ringer and Sensor interaction with the hub and their implementation, these parts will be covered in more detail.

1) The system hub: Two separate processes continuously run on the system hub: unit handler and the web application. Processes communicate between each other in two ways: the unit handler sends messages to the web application by making localhost HTTP requests and the web app uses a FIFO named pipe abstraction to queue notifications for the handler. The web app can send such commands: scan for surrounding BLE devices, fetch new list of sensor or ringer devices, reset all devices, reset all devices and take new base measurements. The handler can request an updated list of sensors or ringers, notify web app of an alarm, connection loss, connection re-establishment events and send BLE scan results. This setup creates two detached, maintainable system.

The unit handler uses multiple-threads and a job queue for distributing specific tasks. The main thread is only used for checking units status, communicating with the web app and queuing appropriate tasks for the worker threads. Separate workers are responsible for tasks such as connection to a device, raising the alarm, issuing reset and re-measure messages. For BLE communication the handler uses Python BluePy library which provides an interface for nearby BLE devices discovery, connection establishment, reading and writing of individual device Service Characteristics.

The hub is designed to continue retransmitting BLE messages to specific connected units until they send back appropriate acknowledgement messages (ACK or RESET_ACK). Such system is employed to counter packet loss in transit and data corruption.

2) Sensor unit: The Sensor unit is a single thread microcontroller, which control flow is displayed in figure 2. The Sensor communicates with the system hub via a separate BLE module which it is connect to with a regular Serial interface. The unit can read and write data directly to and from it avoiding BLE stack complexity as it is abstracted by the HM-10 chip.

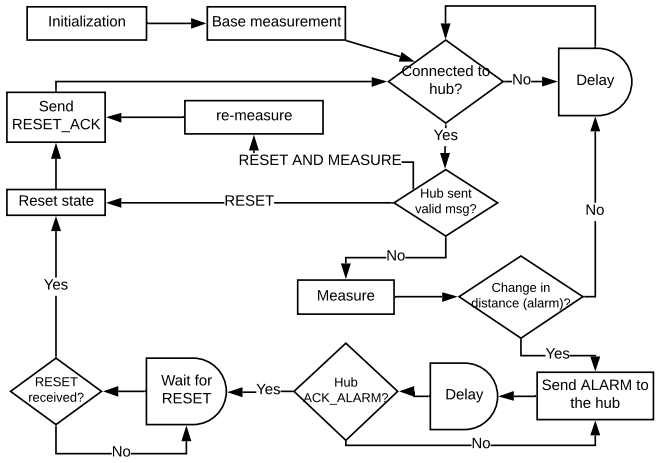

Fig. 2. Sensor unit control flow

Upon start up the Sensor takes an initial ultrasonic sensor measurement which it will compare to other measurements in order to decide whether an alien body has appeared in it's range. The unit then waits for a connection from the system

hub as it is a peripheral device, it cannot initiate a connection. Once connected the Sensor takes new measurements in fixed time periods. If a measurement is not in the delta range of the initial measure, the sensor enters an alarm loop. The Sensor will continuously transmit ALARM messages to the hub until it responds with an ALARM_ACK message, after which the Sensor starts waiting for a restart message from the hub (RESET). The Sensor can also receive RESET_AND_MEASURE command requesting to take a new distance measurement to adapt to the environment alterations.

3) Ringer unit: The Ringer hardware setup is very similar to the Sensor's but instead of the ultrasonic sensor it has a Piezo speaker (buzzer) attached to it. The Ringer control flow is displayed in figure 3.

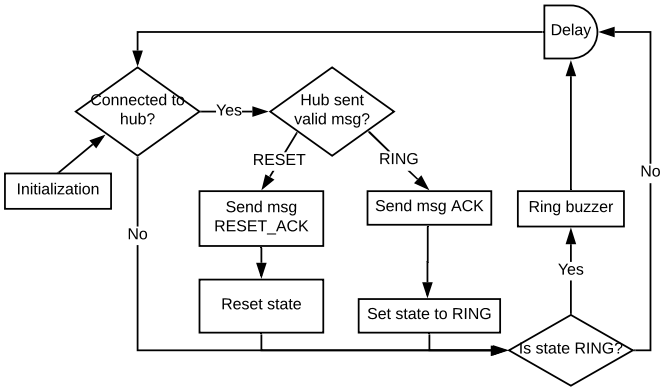

Fig. 3. Ringer unit control flow

The Ringer is configured in such a way that it continues sounding the alarm even if the connection to the hub is lost. It expects to receive only two types of messages: RESET and RING. In both cases, it sends out an acknowledgement message: ACK or RESET_ACK and changes it's internal state accordingly.

IV. Limitations

The current system contains few drawbacks that when resolved would greatly increase user experience.

Even though the system provides close to real time notifications via external email services, they are not immediate. A much more favourable approach would be to send event notifications directly to users smart devices. That would require development of a dedicated app or creation of an app plugin, which would probably be heavily third-party dependent.

At the current setup the cloud infrastructure only supports a single group of users, therefore, a separate cloud instance is required to have private data. The cloud does support multiple users and multiple system hubs registration but information is displayed without distinction between users. This feature was marked as unimportant to the overall system concept, therefore, could be easily implemented in later stages.

In the prototype the user can only reset the alarm system via the system hub and not the cloud application. That creates a situation where a user potentially could not reset the alarm if not being on the same network as the system hub. The problem is partially created due to network configurations and could be quickly resolved by upgrading to an IPv6 network. A solution in NAT environment would be much more complicated.

As the proposed system is a prototype, SSL certificate is not used, therefore, the data is being transferred by unencrypted HTTP. In the deployment environment HTTPS is required to secure networks packets against sniffing.

V. Conclusion

We propose a low-cost wireless alert system that can be easily deployed and configured in residential properties. The extendable system hub uses Bluetooth Low-Energy technology to communicate with any number of Sensor and Ringer units placed around the property. The gathered data is logged and kept off-site in a cloud platform for visualization. The cloud can also process data from multiple system hubs and can be used to edit system subscribers email list. In the event of the alert, users are notified by sound alarm and receive email notifications. A basic user interface provides a convenient platform for configuration and monitoring the systems.

References

[1] Halifax, "What time is it? daylight robbery time.," *Lloyds banking group*, pp. https://www.lloydsbankinggroup.com/globalassets/documents/media/press–releases/halifax/2019/halifax–home–insurance_clocks–go–forward_final.pdf [Accessed 17 May 2019], March 2019.

[2] "Student stuff alert: Digs make rich pickings for thieves." https://www.express.co.uk/finance/personalfinance/858617/freshers-week-2017-university-student-united-kingdom-burglary-theft-illegal-crime [Accessed 17 May 2019], Sept. 2017.

[3] A. Tseloni, R. Thompson, L. Grove, N. Tilley, and G. Farrell, "The effectiveness of burglary security devices," *Security Journal*, vol. 30, pp. 646–664, May 2017.

[4] K. Laubhan, K. Talaat, S. Riehl, M. S. Aman, A. Abdelgawad, and K. Yelamarthi, "A low-power iot framework: From sensors to the cloud," in *2016 IEEE International Conference on Electro Information Technology (EIT)*, pp. 0648–0652, May 2016.

[5] C. Xu, X. Chen, D. Li, and X. Zhong, "Automatic electric meter reading system based on zigbee," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, Oct 2008.

[6] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15.4," in *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 232–237, April 2012.

[7] G. K. Ee, C. K. Ng, N. K. Noordin, and B. M. Ali, "A review of 6lowpan routing protocols," *Proceedings of the Asia-Pacific Advanced Network*, vol. 30, pp. 71–81, 2010.

[8] A. R. D. Kevin Townsend, Carles Cuf, *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*. O'Reilly Media; 1 edition, 2014.

[9] "Iipv4-mapped ipv6 addresses." https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.hale001/ipv6d0031001726.htm [Accessed 17 May 2019].