**Asset Tracking with Magnetometry and WiFi using wireless beacons.**

Nitesh Kumar Salvi

Warsaw University of Technology

**Abstract.** The main agenda of this project was to be able to come up with a novel solution for asset tracking using WiFi nodes and find the absolute orientation and position of the object in space. There are multiple other methods to solve this probelm such as using BLE, RFID, motion sensors and smart tags. The paper presenets a method to accomplish this feat with wifi tags and a hybrid asynchronized network architecture.

**Keywords:** wifi, rssi, magnetometry, esp32, arduino, c++, python, django, elastic search, IoT, wireless sensor networks, embedded, rtos.

**1 Introduction**

The paper presents a complete end-to-end design of an asset tracking system which is capable of finding a persons position in any room of a building and his absolute position in a room approximately and the spacial orientation of the person using WiFi and signal strength (RSSi).

* The hardware consists of a WiFi chip, the ESP32, which can behave both as a client and a server. The chip is capable of hosting its own self-sustained network.
* The system architecture consists of the hardware devices called beacons, centrals and routers. The backend server with a NoSQL database and a frontend written in React.
* The frontend is capable of viewing any person’s latest position and his previous visits to any other room.
* The backend is capable of collecting events and routing requests for the frontend. The backend written in django also has a database connection for storing and retrieving events and filtered requests.
* A visualization tool is made for viewing the person’s position in the room at any given time in history.

**2. Brief System Overview**

1. **Hardware.**

The hardware area of this project is kept quite simple. The system comprises of a Tensillica based 32bit dual code microprocessor with a networking stack. The chip is run with RTOS and has the ability to run firmware written in C.

The chip has an external radio unit and an antenna for radiating WiFi signals as well as BLE, BLE is not used in this project. The chip also comes with a C++ wrapper written for the arduino environment making most of the complex functions encapulated into simpler methods. Functions such as connectTo and disconnectFrom are encapsulated in C++ into much smaller methods which are simpler and also has good exception handling.

The chip can behave both as a client device that connect to any WiFi network, considering it has the required security credentials, it can also host its own access point with **WEP** and **WPA** security, along with hosting its own wifi server on a dedicated port. Multiple sever instances can be created in the script, the chip can handle upto 5 simulateneous client connections at the same time. With soo many of these featuers packed into one small unit this makes it a very powerful and worthy choise for commerical applications such as wireless sensor networks, IoT, swarm systems, robotics, automobile and security systems.

The chip is also equipped with multiple I/O options for interfacing it with external hardware, some of them include I2C, SPI, UART, USART. The chip also has 32 GPIO pins which can be controlled through the arduino framework in C++. The GPIO integration has made the chip quite a lot more powerful giving it diverse options for it to be used in multiple scenarios.

The hardware systems also consists of a magnetometry sensor, the HMC5883L is a 3-Axis digital compass IC, its a high resolution magneto-resistive sensor with a 12 bit ADC that gives it a 10 and 20  heading resolution. It can measure both the direction and the magnitude of the earth magnetic field, from milli-gauss to 8 gauss. It also comes with I2C interface.

1. **Firmware.**

The hardware firmware is written in C++ using the arduino framework made from the C version named ESP IDF, which is the encapsulation of the machine level code written for the tensillica chip.

The C++ version has a host of different libraries giving the user a lot of raw functionalities that can be used to form much more complex network behaviours.

We use the same raw functions to model the network and host an independant network which is hybrid-ascynchronous. The term hyrbrid here is because, the nodes communicate with each other in a synchrnous fashion upon reception of information from the beacon. The firmware is written in an orderly fashion of handling events asychroniously and sending back consistent responses, for user actions.

There are two different firmware segments, one for the central units and one for the beaon units.

* **Central Firmware**

The central firmware is again sub-divided into two portions, the **sentinal** node and the **slave** node.

1. **Sentinal Node**

The sentinal node is considered as the gateway node. Which intercepts information from the slave node. The node also runs a DHCP server on a hotspot with a static IP and port. Each central nodes has a unique ID assigned in code, both the slave and sentinal share the same ID, since connections from adjacent rooms shouldn’t be intercpeted, this is basically whitelisting a set of entries of ID’s and in this particular case the whitelist is limited to only one ID. This also makes the firmware exteremely generic and reusable.

1. **Slave Node**

The slave node routes the information received from the beacon to the sentinal node, which routes the packed information to the central server and is packed on the database. The slave runs a DHCP server with a hotspot on a static IP and port. When the slave receives a connection from the the beacon, it performs a synchronous connection action, where it disconnects from the client, and connects to the sentinal node and transmits the information relative to the slave and goes back into idle mode, this action takes places with a set delay period and the delay period choice has a lot of impact on the accuracy of the positioning.

* **Beacon Firmware**

The beacon is an independant and stateless system, which can connected and disconnect to a network at timed intervals ( at will ). The beacon has a stack for behaving as a client device for a DHCP server and dynamically allocated IP address.

The beacon also bears a wrapper for the I2C layer which can communicate to the HMC chip for polling magnetic data at discrete intervals of time.

1. **Server software**

The server software is written in Python, and is chosen due to its large userbase and abbundance of resources with relating to web development.

Django is a webframework that needs no introduction, the framework can host, route and serve webpages on a single instance, making it extremely user friendly and horizontally scalable. The framework can also be integrated with 3rd party databases which can be either SQL or NOSQL. Django can handle requests on a certain URL might it be POST, GET, UPDATE, PUT or DELETE. It follows the norm of the MVC architecture, where certain views are sent as responses for certain requested routes. This makes is integrateable on many levels.

In our case, the sentinal node connects to a central WiFi which can send POST requests with the contained beacon information to a certain route, this would trigger a certain view method to be called and the body containing the information is parsed, cleaned, verified and dumped to the database through the database handler. This makes the entire architecture soo much more simple, eliminating the use for a broker for parsing the symmetrizing the data.

Most wireless sensor networks employ a broker which mediates the hardware to the central sever, the broker behaves as an encapsulation inbetween, making itself similar to a computer client sending requests to the django server and receiving views in return. The returned response would have status codes and acknowledge messages for stating successful network actions and continuing the process smoothly and also maintaing RESt standards.

The static files such as HTML, CSS and JS along with assets are hosted on a separate server written in React.Js hosted via CDN. This making the backend completely independant and can be reverse proxied for scaling the system for 100’s of users.

1. **Frontend Software**

The frontend being the most captivating, renders views and actions in a pleasing and aesthetic way for users to comprehend, this is written in React.JS, react has been chosen for its virtual DOM properties, making it ideal for notification reliant systems, that can spawn and render views without rendering the whole page at once.

The entire frontend is divided into separate components which can be pivoted and used in multiple ways, also making the project resuable and highly maintainable.

1. **Principle Working Algorithm**

The entire system consists of these key components.

1. Beacons ( B )
2. Central – Sentinal ( CSN )
3. Central – Slave ( CS )
4. Backend – Django (BD )
5. Backend – Elasticsearch ( BE )
6. Frontend – React ( FR )

Each component in a single entity in itself which has been combined together to form working model of an asset tracking sytem.

**Table 1.** Workflow of the network behaviour and the behaviour of the hardware components.

|  |  |  |
| --- | --- | --- |
| **Action Index** | **Description** | **Next Point** |
| 1. | BS and CS *x*  meters apart with matching ID. | 2 |
| 2 | B walks into room, and connects to CS, transmits RSSi, orientation and ID. Keeps record of current connection made. | 3 |
| 3 | CSS parses information received, packs them into JSON and transmits the packet to CSN by connecting to its hotspot with SSID and password. Connects after *t+DELAY* seconds after receiving the packet from B. | 3.1 |
| 3.1 | CS disconnects from CSN and gets back into idle mode awaiting new connections. | 4 |
| 4  5 | CS receives new packet from B, parses it and reparses packet from previously received CS and connects to central WiFi network to transmit packet to the Django server.  CSN has recived corrupt data from either BS or CS, it sends the partial data to ther server, or if both are corrupt, it discgards packet. | 5  1 || 6 |
| 6 | Django receives POST request to a specified URL and has the beacon information as body elements. Django checks for authenticity of the packet with the ROOM ID from where it came and the beacon ID. If its not authentic then the packet is disgarded and an error response is sent back. | 7 |
|  |  |  |
| 7 | React fetches information based on the selected filters and sends the parsed information with correct response to the client. | 8 |
| 8 | Received filtered response is time based, and a unique mapping tool is used to translate the distances of the beacons to absolute room positions. The orientation is used to show the direction of the person standing with respect to the magnetic north of the room. | 7 |

**4. References**

[1] Prof. B Nithya Ramesh, Aashay R Amballi, Vivekananda Mahanta, Django The Python Web Framework, Department of Master of Computer Applications, NHCE, Bangalore, India.

[2] Isable Drost-Fromm, Software Engineer, Elastic, Testing Automation for distributed Applications.

[3] FALK, R. and HOF, Security design for industrial sensor network Information Technology, Vol.52, No. 6, Oldenbourg, 2010.

[4] KALITA, H. K. and KAR, A Key management in secure self-organized wireless sensor network: a new approach, Proceedings of the International Conference and Workshop on Emerging Trends in Technology (ICWET ‘11). ACM, 2011, pp. 865870

[5] JHA, M. K. and SHARMA, T. P, Secure data aggregation in wireless sensor network: a survey International Journal of Engineering Science and Technology (IJEST), Vol. 5, No. 3, 2011.