XPark:

A low power Wireless Sensor Network for automatic parking

Addarsh Chandrasekar

Source code: https://github.com/Addarsh/XPark

Overview

Parking is a known problem in major urban areas in the world. It is both time-consuming and stressful to find a good parking location close to the desired destination. Often times, drivers have no assistance in finding a good parking spot and therefore waste time and fuel driving around to find an open parking spot. With the population of urban cities growing dramatically and the number of parking vacancies not keeping up, finding parking will only become a bigger problem ultimately leading to traffic congestion and inefficiency.

The ideal solution for the parking problem is to know the occupancy of each parking spot in real time (low latency) and to be able manage parking spots by notifying and directing vehicles as they move in and out of these spaces. The parking management system would then be able to seamlessly manage incoming and outgoing vehicles thus drastically improving the efficiency of the parking process and urban transportation in general.

To realize this solution, I implemented a wireless network with one sensor node on each parking lot that combines Bluetooth Low Energy and Low Power Wide Area Network to perform real time occupancy detection of parking spots. Many other solutions like using cameras have been proposed for automated parking management systems before (cite camera paper etc.) and while these solutions make sense from an infrastructure cost and maintenance perspective, they also have associated disadvantages. Camera based solutions assume that there exist sufficient infrastructure in the form of walls and power outlets to mount cameras such that an entire parking lot is covered. In reality, this solution may work for some parking spots but for the majority of outdoor parking spots, it is hard to fit all these assumptions. Despite the fact that computer vision accuracies have greatly improved with the advent of deep learning algorithms, very high accuracies are very much dependent on environmental factors such as good lighting and minimal occlusions.

Topology

The prime objective was to come up with a low power solution for automated parking. If each node of the system consumes a lot of power, then the solution cannot scale.

From my initial research, BLE was the lowest power wireless technology for PAN and LoRAWAN was the lowest for power wireless technology for Wide Area Networks.

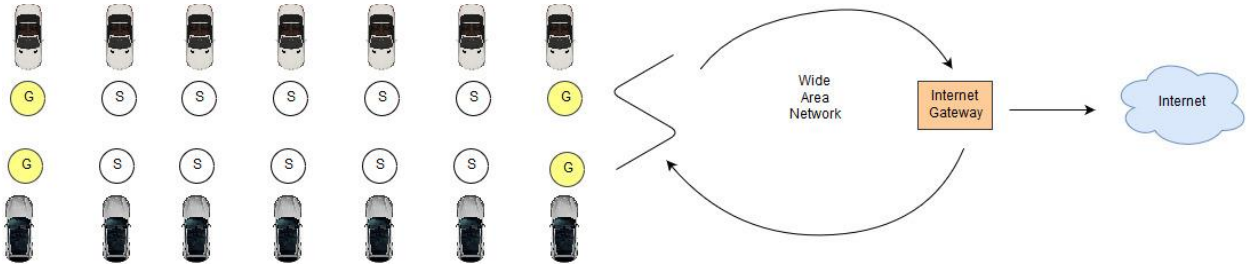
Here are the brief highlights of BLE and LoRa:

BLE – small TX and RX currents (~10 mA), medium range (10-50 m range)

LoRa – high TX and RX currents (~100 Ma), long range (up to 20 km range)

I decided to mesh both technologies to combine their best strengths to get the desired result of a low power solution with maximum range.

Here is the topology of the network:

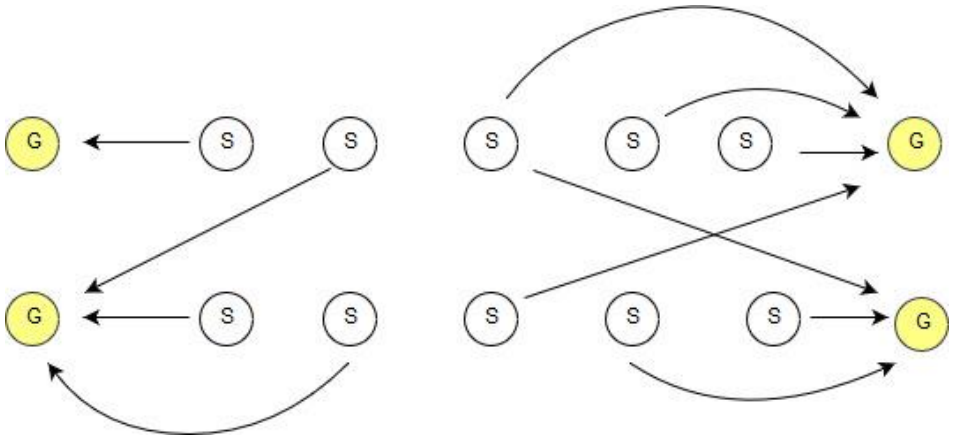


Each parking spot consists of one sensor node to detect occupancy. The overall network can be broken down into two sub-networks: Personal Area Network (PAN) and a Wide Area Network (WAN). Bluetooth Low Energy (BLE) is used for communication within the private network while LoRa Wide Area Network (LoRaWAN) is used for the wide area network. The combination of two low power technologies is used to minimize the power consumption of each sub-network and thereby the power consumption of the overall network.

Sensor nodes within the network can be classified into two types: Gateway node (denoted by G) and Slave Node (denoted by S). Gateway nodes are responsible for connecting the PAN to the WAN in addition to monitoring the occupancy of their own spot. Slave nodes have the responsibility of only monitoring the spot status and communicating information to the corresponding master node.

Personal Area Network

The Personal Area Network consists of nodes that communicate with each other via Bluetooth Low Energy. Within the PAN, the “hub and spoke model” is used for the communication among nodes. The gateway nodes serve as hubs with the slave nodes forming the links/spokes. Each slave node looks to pair with a gateway node that is closest to it in terms of Receiver Signal Strength Index (RSSI) value. The RSSI value of the gateway is used as an indication of the possible strength of link strength upon pairing.

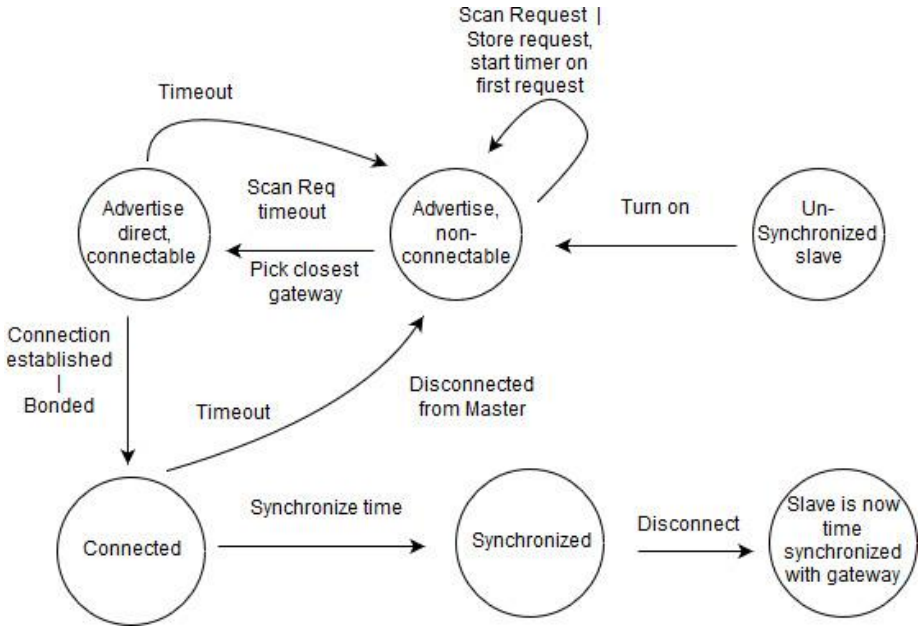


Personal Area Network (pairings)

Node pairing algorithm

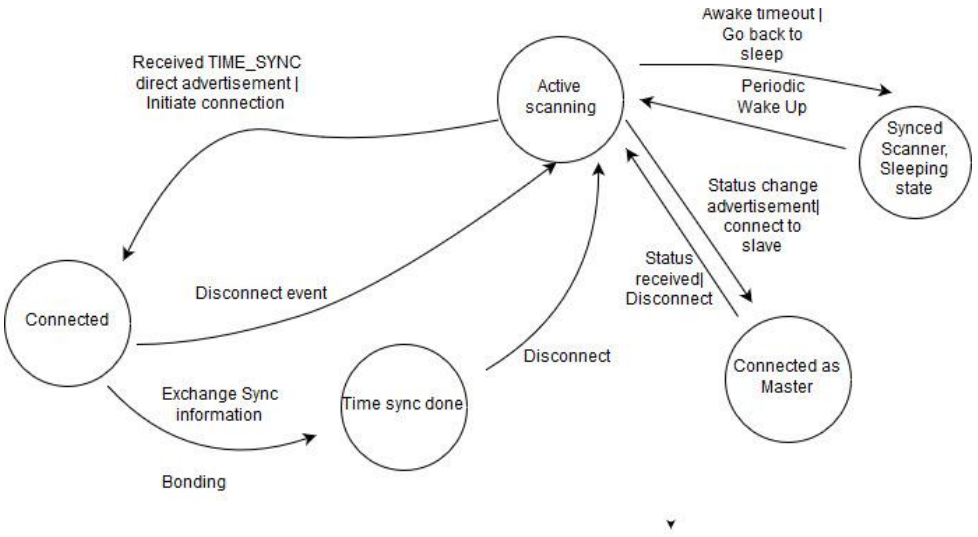
Slave nodes out of reset are unsynchronized nodes looking to pair with the closest gateway node. The pairing refers to not only making a connection, but also exchanging synchronization with the local time of the gateway node. On the other hand, each gateway node out of reset looks to pair with and synchronize with slave nodes that are still unsynchronized and unpaired.

Let us the take an example of 1 gateway node with 3 slave nodes that are unsynchronized. The state of the unsynchronized slave evolves as follows.



The slave starts off advertising non-connectable, to reduce transmit power and avoid possible wrong connections. As it advertises, the slave receives scan requests from potential gateways and after a fixed time, it picks the closest gateway and starts to directly advertise to it. It is here that the selected gateway is expected to connect to this slave. If connection occurs for the first time, then the connection leads to bonding. Once this is complete, the devices are disconnected and the slave is now synced with the gateway.

The gateway state machine for the node pairing algorithm can be seen as follows.

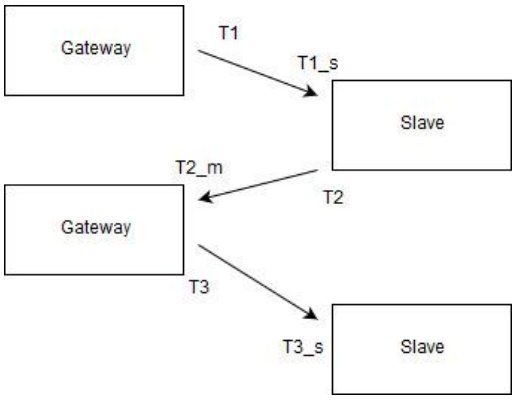


The gateway is periodically scanning for unsynchronized slaves. When it finds one, it tries to connect to the slave and time synchronize with it. Once it synchronizes all nearby slaves, it wakes up periodically to check for spot status update advertisements from the relevant synchronized slaves.

The data received from the slaves is then sent over to the receiver via the Wide Area Network which then connects to the internet.

Time synchronization

The time synchronization of the slave with the master is completed using Precision Time Protocol. After synchronization, the slave's local time will be equal to the gateway's local time (ideally within 100 microseconds accuracy). The message sequence is as follows:

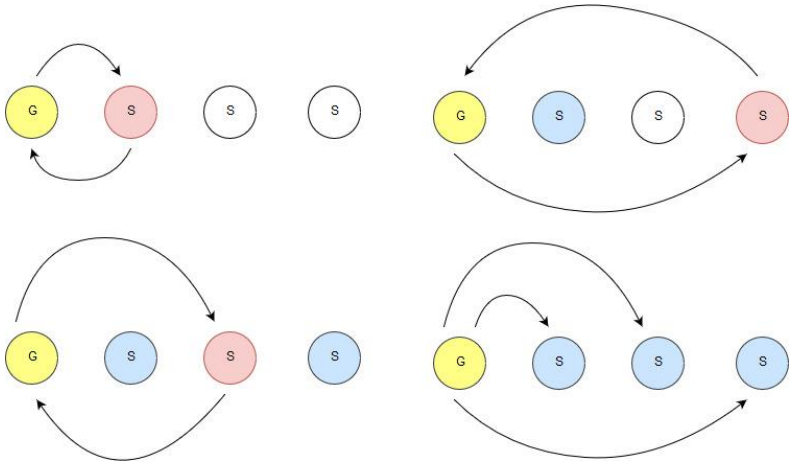


The clock offset calculated by the slave is given by:

$$\sigma = (T1_s - T1 - T2_m + T2)/2$$

The slave corrects its local clock using the offset and immediately disconnects. The gateway and the node are time synchronized and can wake-up at the same time in the near future.

The gateway performs synchronization with several slave nodes in a manner seen in the picture below



Synchronization process seen sequentially from top left to bottom right.

Wide Area Network

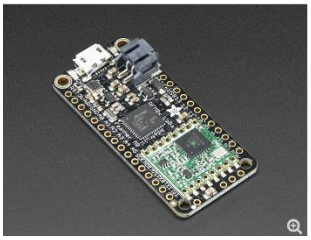
The Wide Area Network uses LoRa Wide Area Network at 915 Mhz to perform the network transmit and receive. LoRaWAN is characterized by High transmission range of up-to 20 km line of sight by increasing the receiver sensitivity at the same transmit power. Since our requirement is to use low power and achieve a wide range, LPWAN is the best fit for our application. For this application, a LoRa chipset (RFM95) is used to send and receive data to the internet gateway.

Hardware

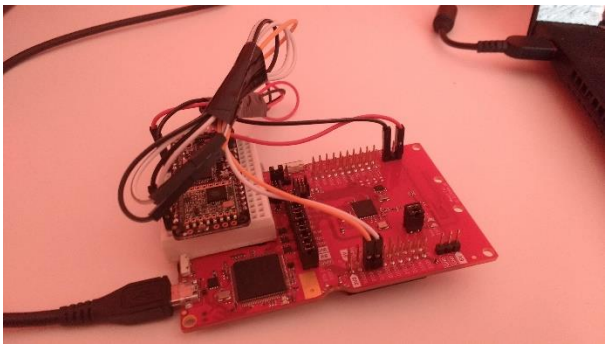
For this project, I used the TI CC2640R2F launchpad which comes with a BLE 5 networking stack built in. I used an infrared sensor OSEPP IRDET-01 for the occupancy detection and Adafruit's feather M0 board with RFM95 for the LoRa module.

TI CC2640 R2F

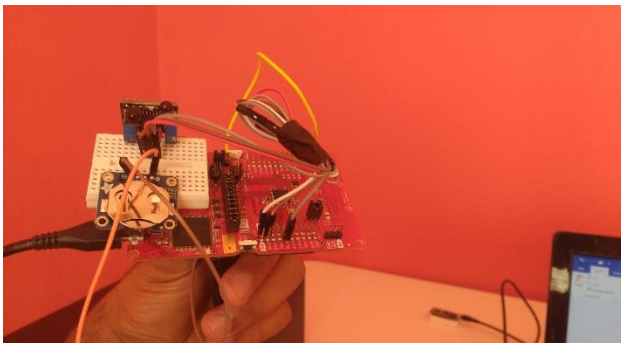
OSEPP IRDET-01



Adafruit feather M0

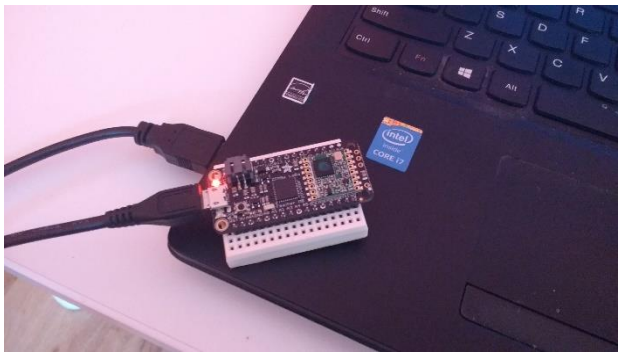


Gateway Node



Slave Node

Slave node in action



Internet Gateway

Results

The gateway and slave are synchronized to wake-up and talk to each other every 20 seconds. Thus, after every 20 seconds, the slave node wakes up, checks if the spot has updated and then starts advertising if there is a change. If not, the node goes back to sleep again. Similarly, the gateway node wakes up at the same time and scans for advertising slaves for 2 seconds. If no slaves, are found within 2 seconds, the gateway goes back to sleep until the next wake-up period. If a slave is found, it connects securely to it and receives the updated information. The updated information is sent across to the LoRa receiver gateway which then displays the status.

Here is a picture of the parking spot update being sent when the status of the spot changes.

```
LoRa radio init OK!
Set Freq to: 915.00
Parking status Update: SpotNumber 1 is now open!
Parking status Update: SpotNumber 1 is now occupied!
Parking status Update: SpotNumber 1 is now open!
Parking status Update: SpotNumber 1 is now occupied!
```

Power consumption analysis

Each slave node wakes up every 20 seconds, checks if the spot status has been updated and if so, sends the updated status to the gateway. The whole process takes about 1s of scanning/advertising and 3 seconds of connection time for transfer.

The total power consumed by the slave in such an event (9 mA advertising current, 6 mA connection current) = 9*1 + 6*3 = 27 mAs.

We assume that the spot occupied by the slave changes around 15 times on an average in a day.

In such a case total power consumed in a day by the node = 15*27 mAs.

With a battery capacity of 1000 mAh, the total number of days the slave node would last

= (1000*3600)/(15*27) = 8888 days = 24 years.

For a gateway node, it has to wake up every 20s, scan for 2 seconds, connect if needed and go back to sleep.

On a average, one can assume 100 connections events in a day. On each connection, the LoRa module transmits for about 100 ms at 125 ma.

In addition, the gateway wakes up every 2s for 20 seconds. (at 6 ma current for scanning).

The total power consumption = 3600/20*2*6 + 100 *(6*4 + 12.5) = 90*24 + 100*24 + 1250= 190*24 + 1250 mAs = 5810 mAs

With a battery capacity of 1000 mAh, total number of days the slave node would last:

= (1000*3600)/(5810) = 612.612 days = 1.68 years.