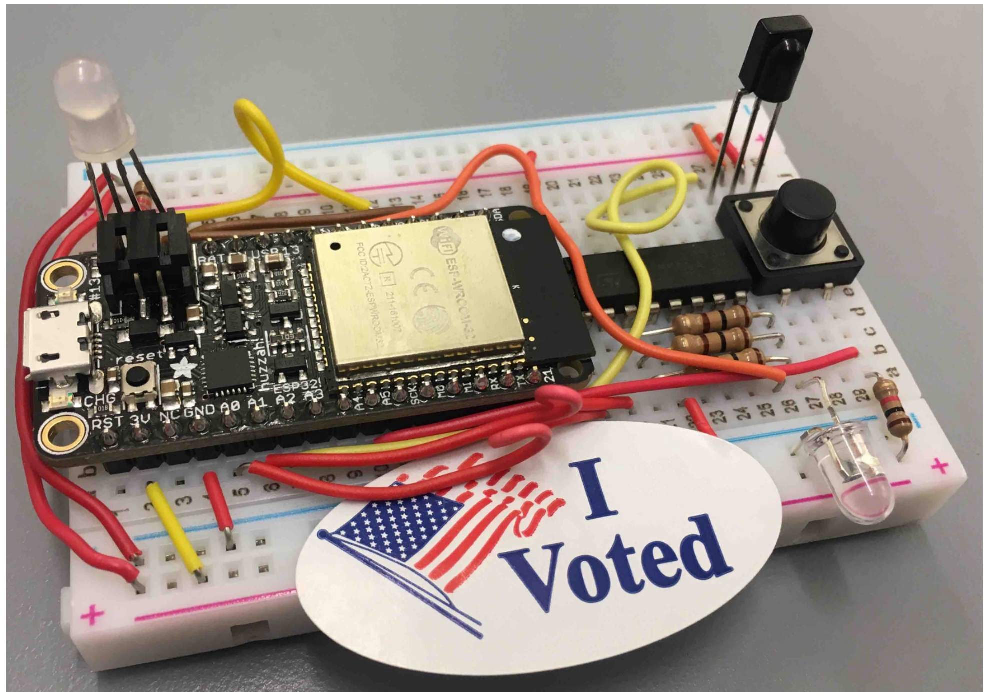


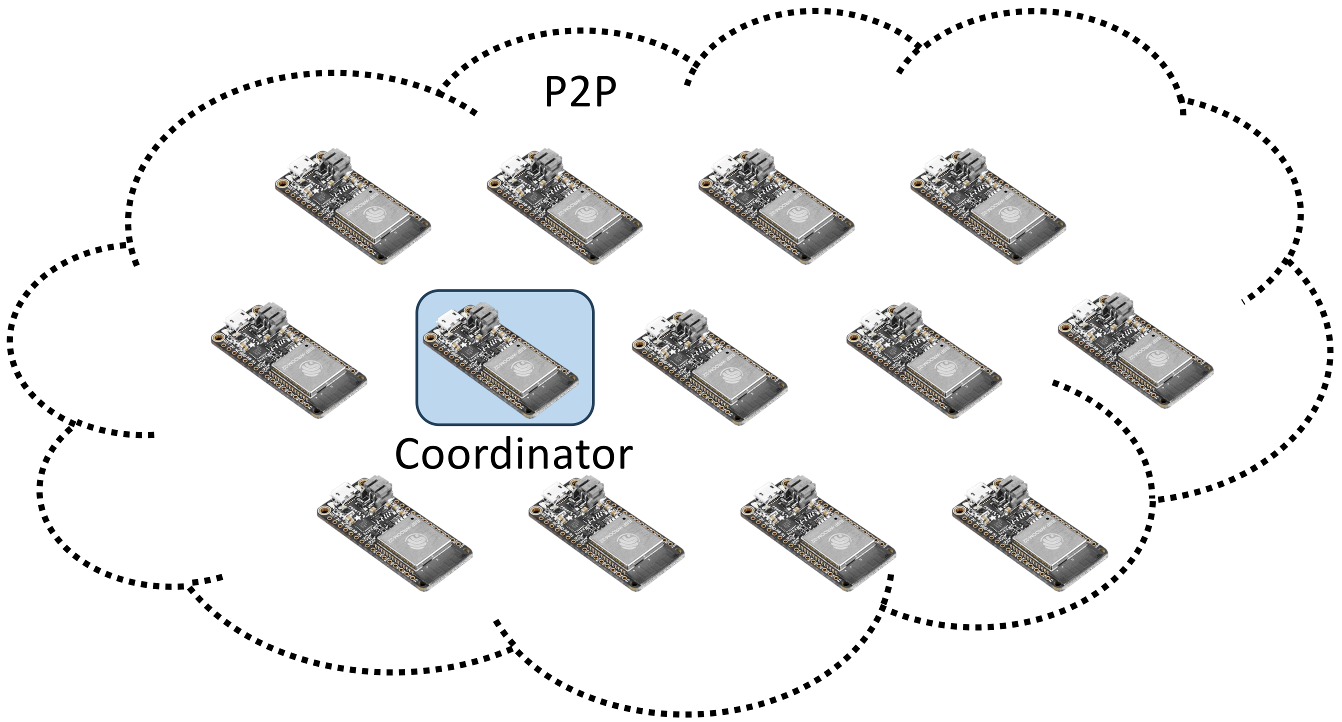
BU-EC444

Fall 2024
Prof. Little

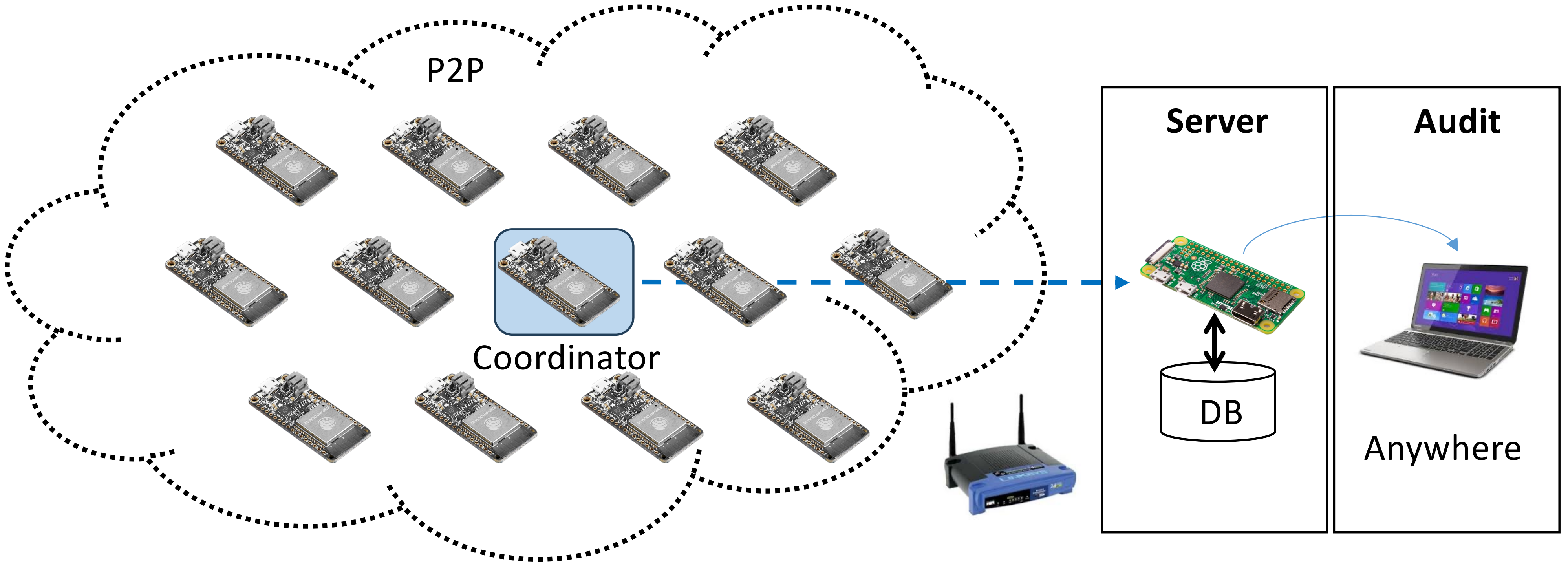
Review of Quest: Coordinator Election

Quest 5: Coordinator Election

How to pick one unit to be coordinator in a wireless distributed system?



Quest 5: “Coordinator” Election

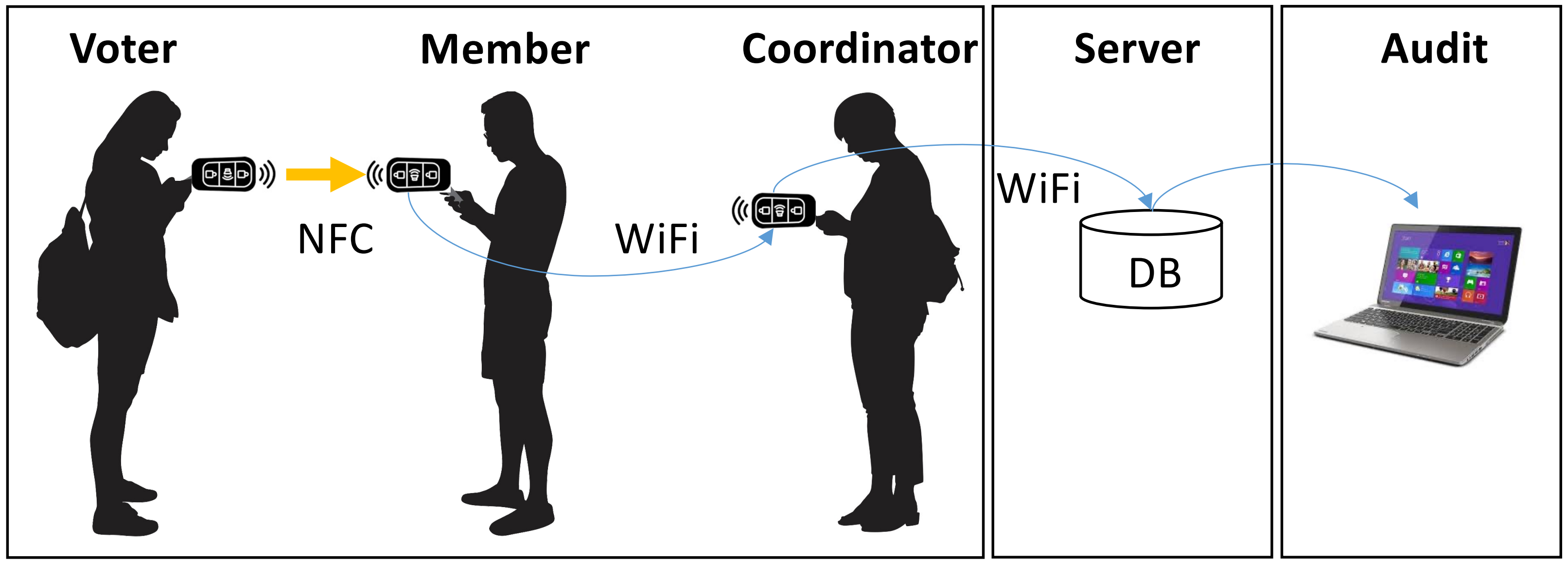


- The network is peer-to-peer: address messages directly to peers
- The coordinator is the only unit that talks to the server
- The coordinator needs to be replaced if it goes away
- One and only one is deemed to be “coordinator”

Quest 5: System Rules

General Rules:

- Can only vote if you connect via NFC first
- Subsequent connections are WiFi
- One of the members will be be elected coordinator



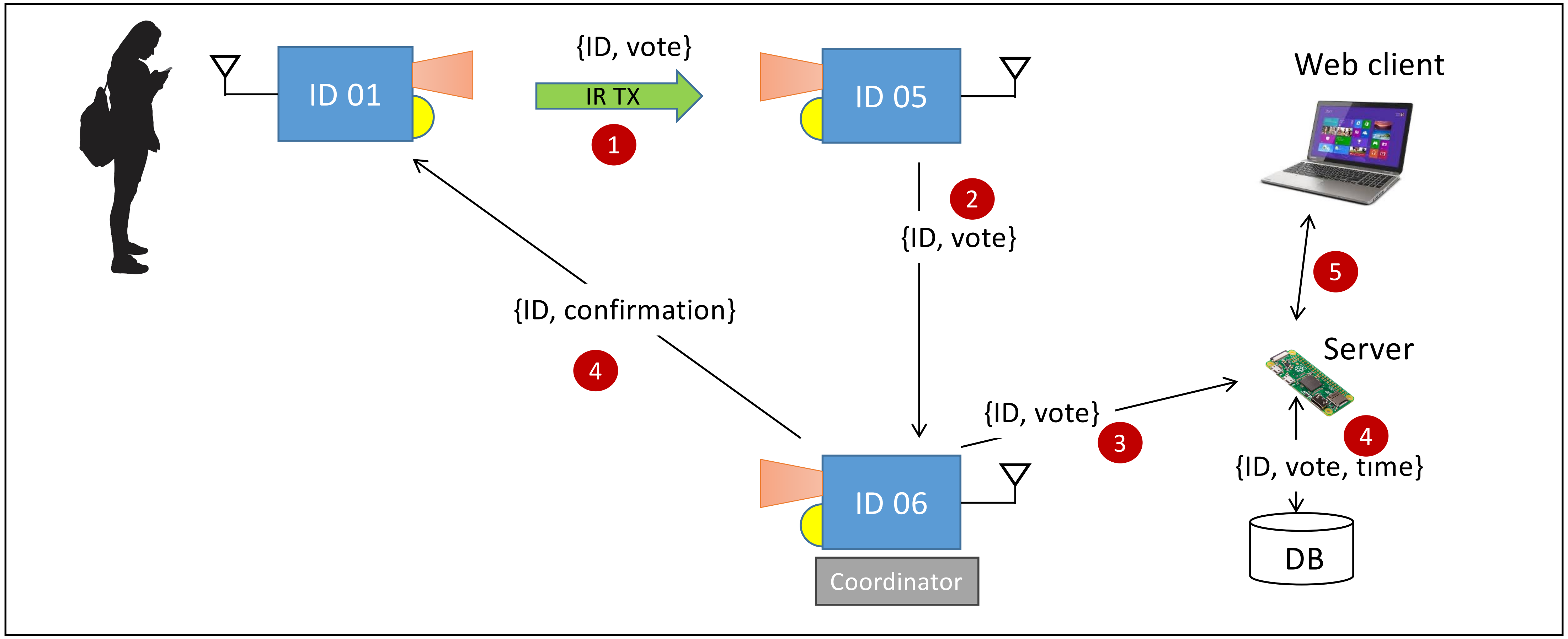
Key Takeaways

- Near Field Communications
- Wireless security
- Peer-to-peer
- IoT provisioning
- Databases

Voting Rules:

- Voter passes ID to one of the members
- Member sends ID and vote to Coordinator
- Coordinator sends the ID and Vote to the database server
- Coordinator sends confirmation to original voter fob

Quest 5: Voting in the System Expanded



Problem decomposition

Part 1

1. Build your fob hardware (all units identical)
2. Implement a coordinator election algorithm
3. Demonstrate

Part 2

1. Build out the voting process
   - IR TX/RX transmitting from Fob to Fob
   - UDP message sending from Member to Coordinator
   - UDP message sending from Coordinator to Audit Server and confirmation to fob (WiFi)
2. Create a node server to handle database logging ID, vote, time
3. Create a web interface to display data logged in server database

Approach

1. Assume you know the units on your local subnet
2. Using WiFi and the known device addresses, initiate coordinator election for all available devices on boot
3. Elect a coordinator using the election protocol
4. The coordinator becomes the entity that talks to the database server
5. Do voting using the NFC data exchange and registration
6. Audit behavior using a web browser

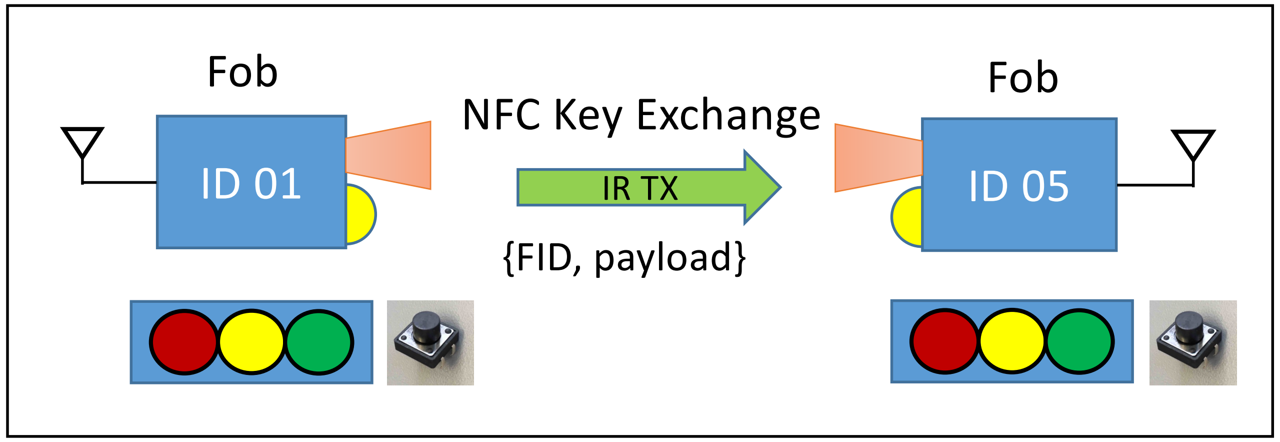
Quest 5 Cluster

- IR TX/RX
- Databases
- State Models
- Coordinator Election
- Security, provisioning

Skill: IR TX/RX

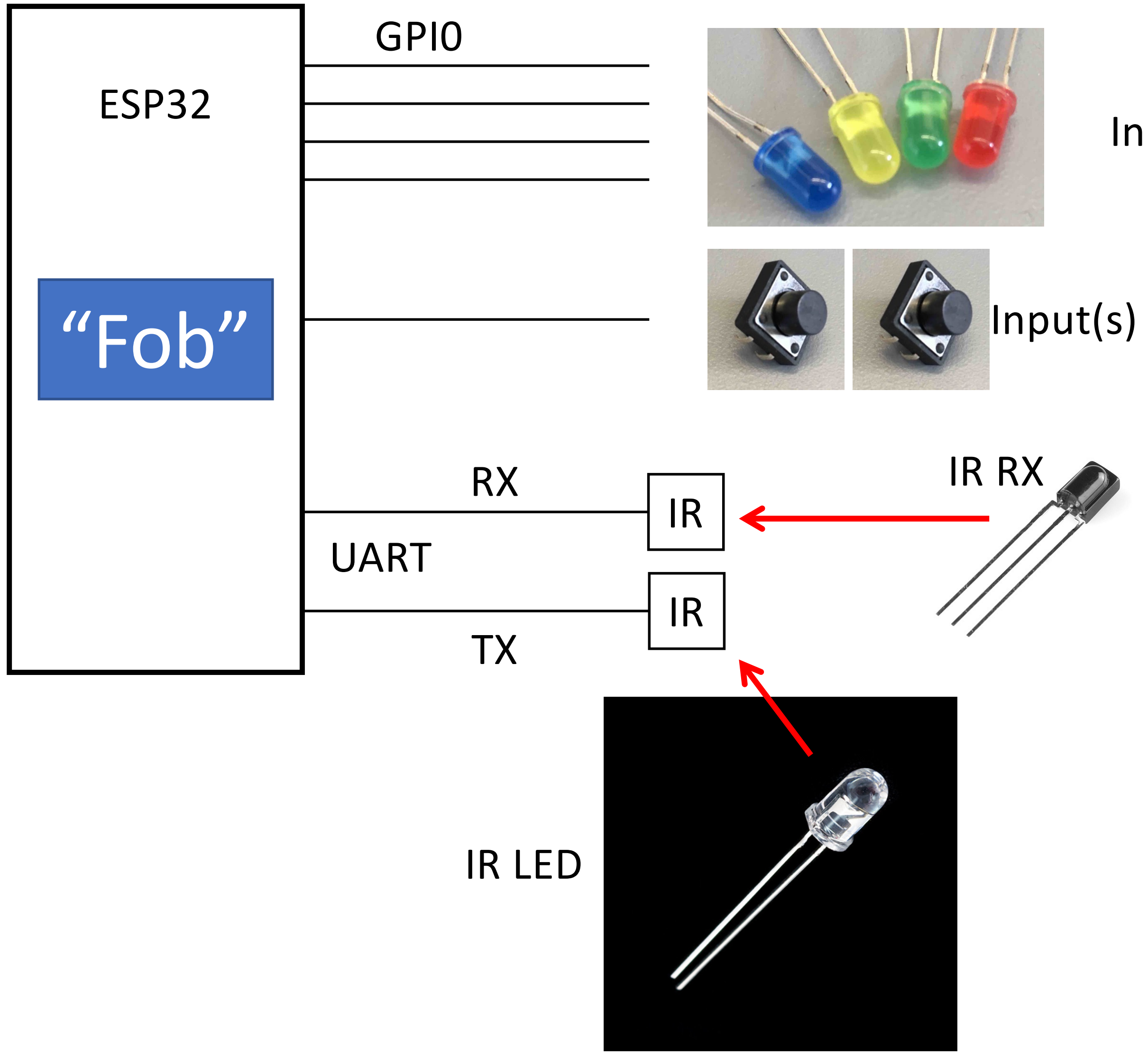
Near Field Communications – IR TX/RX

Parts – the IR NFC data exchange

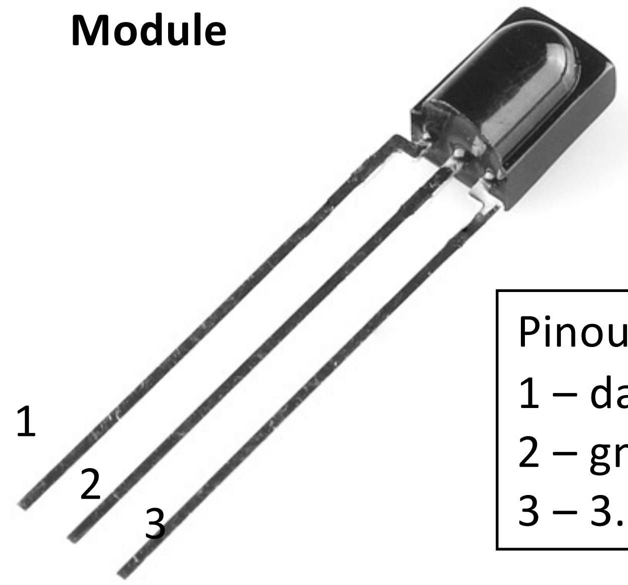


Bring this up first!

NFC with IR – similar to TV remote control

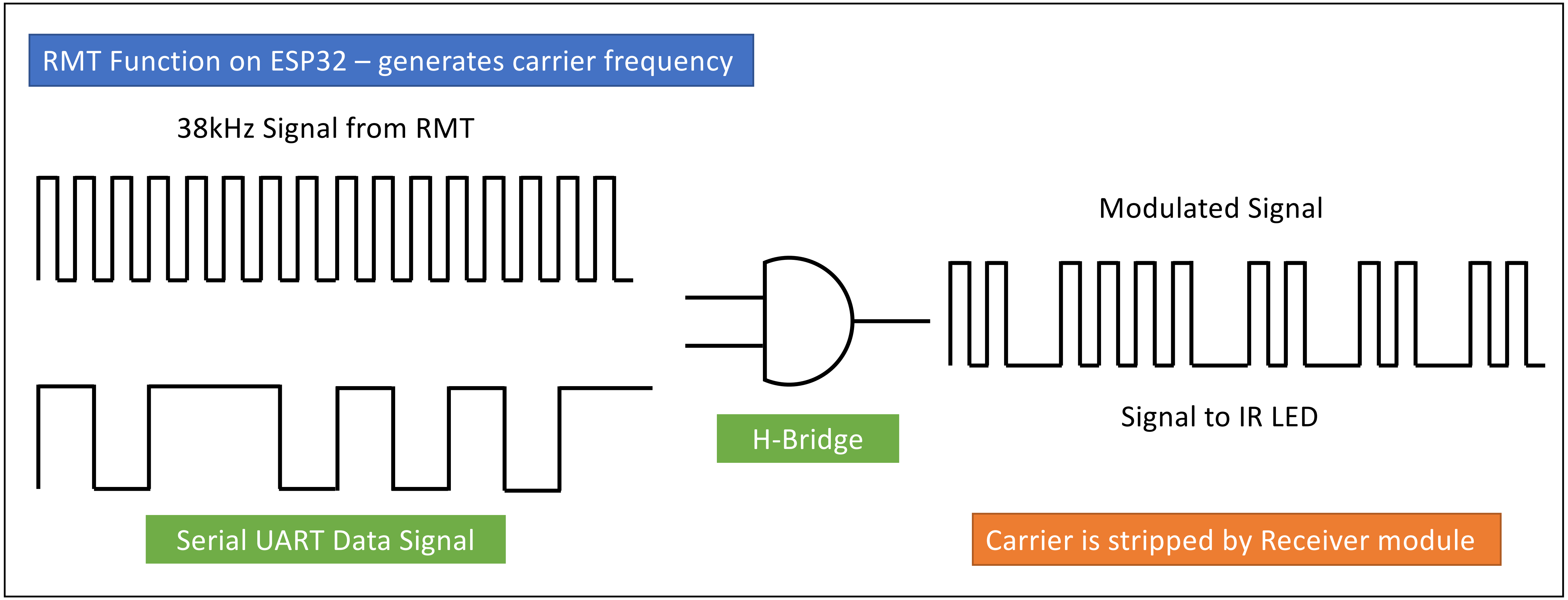


TSPO382 IR Receiver Module



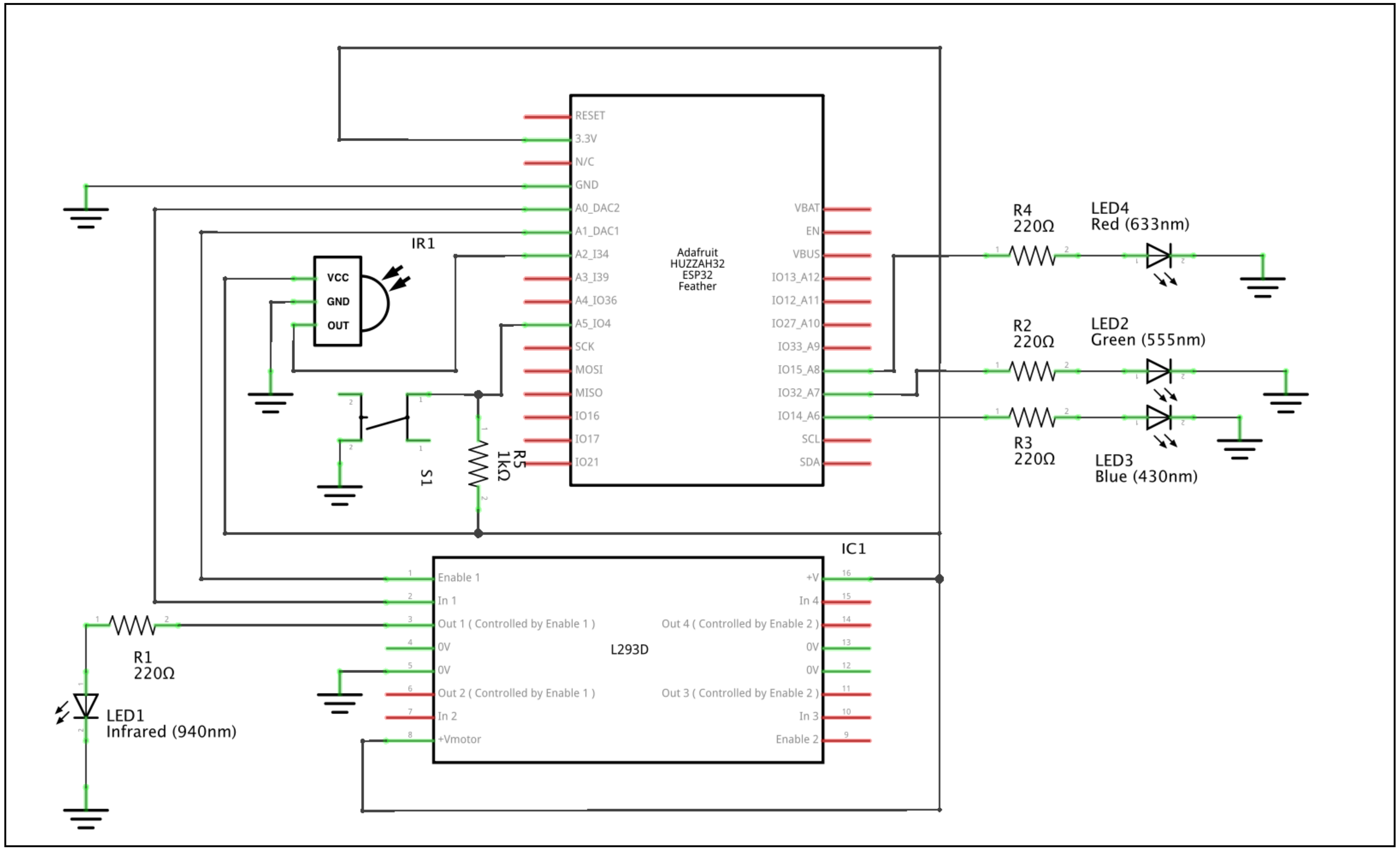
Pinout
1 – data out
2 – gnd
3 – 3.3V

Modulation of carrier signal with UART data

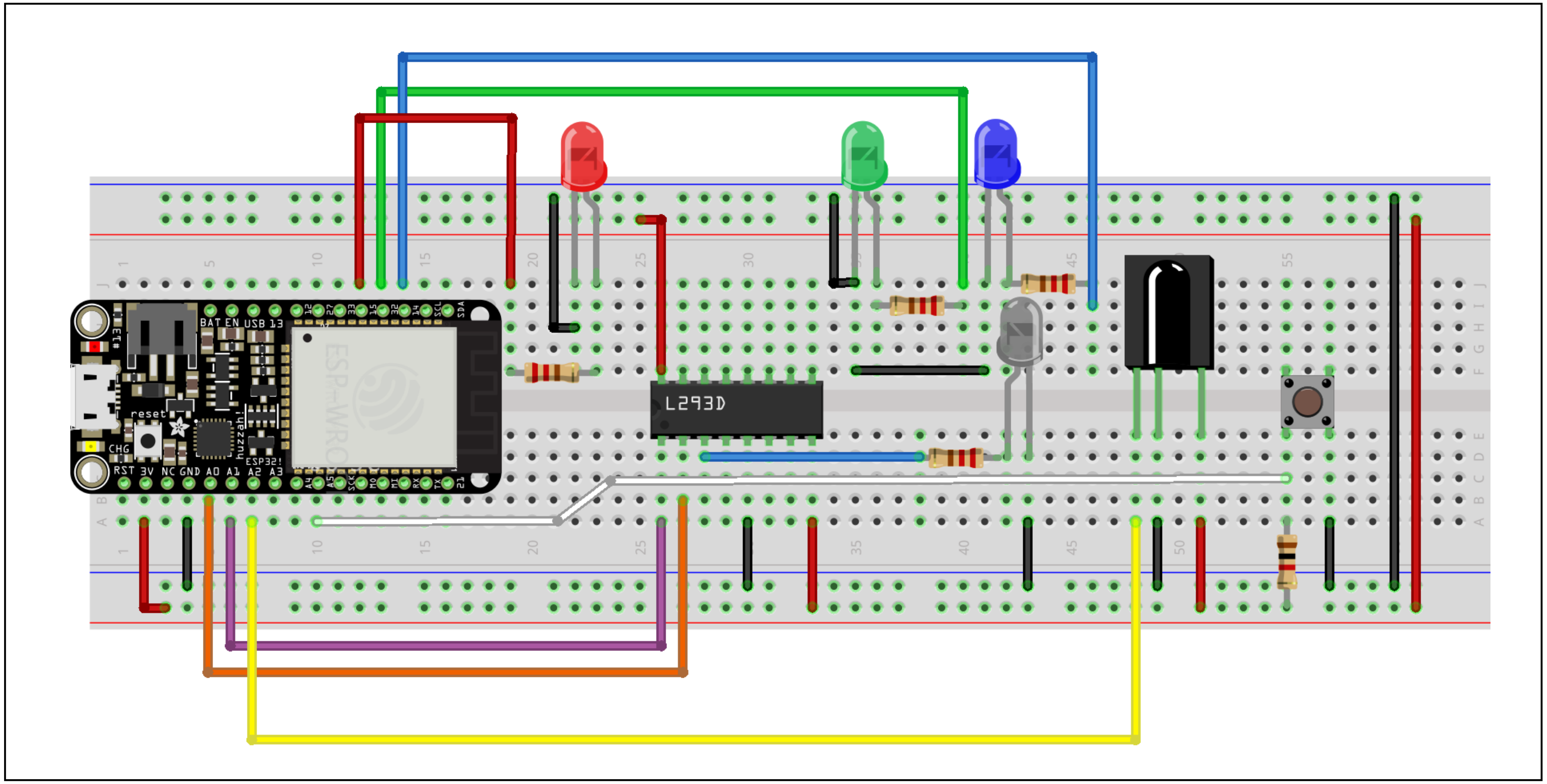

RMT Function on ESP32 – generates carrier frequency
38kHz Signal from RMT
Modulated Signal
Signal to IR LED
H-Bridge
Serial UART Data Signal
Carrier is stripped by Receiver module

NOTE: we use RMT to generate carrier only
Then UART to decode received signal

Schematic


IR1
VCC
GND
OUT
RESET
3.3V
N/C
GND
A0_DAC2
A1_DAC1
A2_I34
A3_I39
A4_IO36
A5_IO4
SCK
MOSI
MISO
IO16
IO17
IO21
Adafruit HUZZAH32 ESP32 Feather
VBAT
EN
VBUS
IO13_A12
IO12_A11
IO27_A10
IO33_A9
IO15_A8
IO32_A7
IO14_A6
SCL
SDA
R4 220Ω
LED4 Red (633nm)
R2 220Ω
LED2 Green (555nm)
R3 220Ω
LED3 Blue (430nm)
S1
R5 1kΩ
IC1
L293D
Enable 1
In 1
Out 1 (Controlled by Enable 1)
0V
0V
Out 2 (Controlled by Enable 1)
In 2
+Vmotor
+V
In 4
Out 4 (Controlled by Enable 2)
0V
0V
Out 3 (Controlled by Enable 2)
In 3
Enable 2
R1 220Ω
LED1 Infrared (940nm)

Fritzing



L293D
ESP32!
huzzah!
reset
BAT EN USB 13
RST 3V NC GND A0 A1 A2 A3
CHG

Assignment

- Build a fob for each team member
- Test with provided baseline code (“ir-txrx-example” code)
- Adapt code as per skill assignment (**Note: must enable TX or RX in the code**)

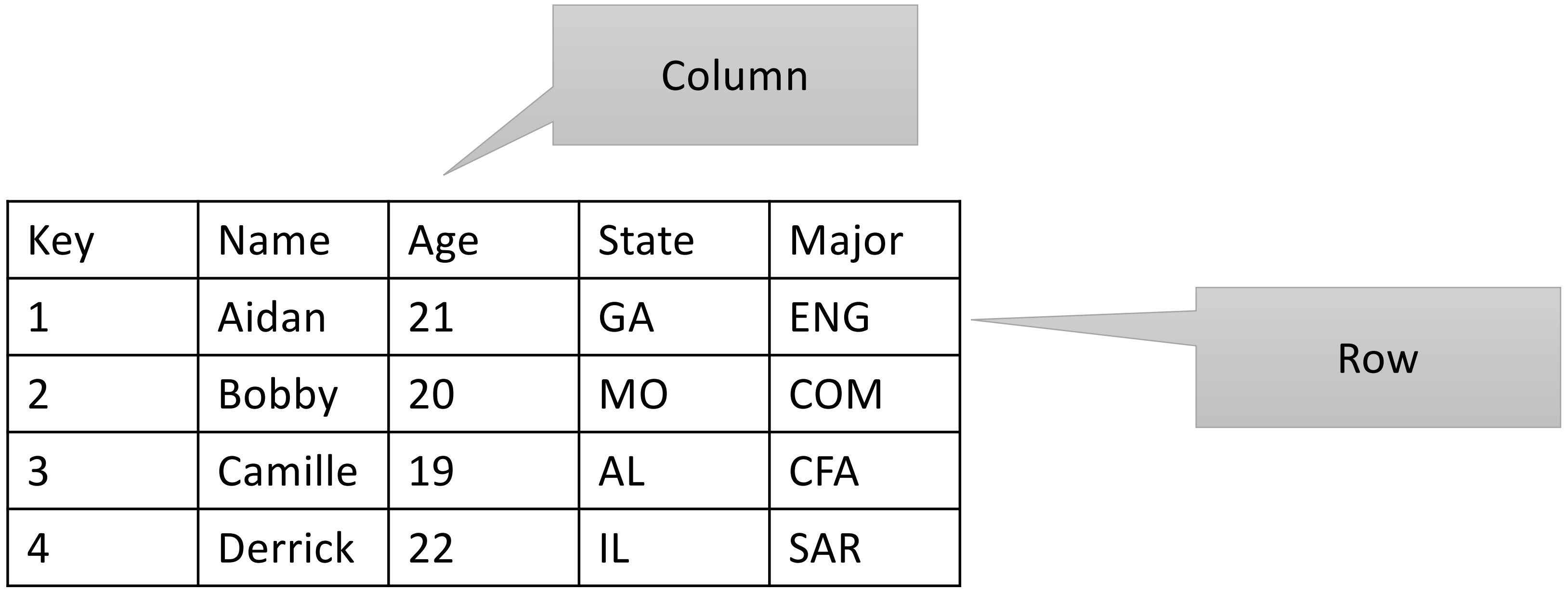
Resources

- UART – serial communications on ESP32
- ESP MC PWM used to generate clock on enable pin for the H-bridge driver (set at 50% duty cycle and fixed pulse width)
- These are logically combined (AND)
- On the receive-side, receiver part eliminates the clock – direct to UART

- Links:
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/mcpwm.html
  - https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/uart.html

Skill: Databases

Databases



Key	Name	Age	State	Major
1	Aidan	21	GA	ENG
2	Bobby	20	MO	COM
3	Camille	19	AL	CFA
4	Derrick	22	IL	SAR

Concepts

- Table-oriented
- Relational
- Key-value pairs

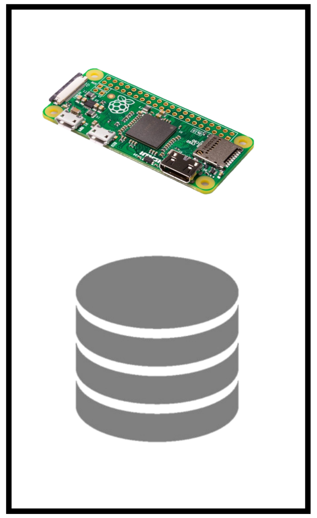
Queries

- Return one or more rows or reduced rows based on input selection

Persistence

- Data is saved through power failure

Consider a local server to log indoor climate data (“edge server”)



Smart Space

Databases (or database management systems)

- Environmental data example

  (**Sensor ID**, sensor location, sensor calibration data, sensor software version, time, temperature, pressure, humidity, CO2 level)

- Put into a **log file** (e.g., CSV) or
- Put into a **database** (with database interface)
- Log file: simple and effective, but harder to find something
- DB/DBMS: better for selecting a subset of data
- Examples (queries)
  - Give me all records with a value range of Temp > 30 (**selection**)
  - Give me all sensor locations in the system (**projection**)
  - Give me all sensor locations where temp > 30 (**selection** and **projection**)

Databases (or database management systems)

- Environmental data example

  (**Sensor ID**, `sensor location, sensor calibration data, sensor software version, time, temperature, pressure, humidity, CO2 level`)

- Also: efficiency by separating data into multiple tables
  - Table 1: (**Sensor ID**, `sensor location, sensor calibration data, sensor software version`)
  - Table 2: (**Sensor ID**, `time, temperature, pressure, humidity, CO2 level`)
- This allows reducing redundant information in the tables.
- The recombination function is called a **join** and works on the **key** attribute which is common on both tables

Database Skill

- Bring up a tingodb database on your laptop
- Interface with node.js
- Demonstrate with smoke.txt file (a static example)

See

- Design pattern for databases
- Tingodb: http://www.tingodb.com
- Mongodb https://www.w3schools.com/nodejs/nodejs_mongodb.asp

In the quest you will use a live example using the esp32-sourced data as input

Node and tingodb – example logging sensor data (inserting data)

Create a folder for your database:

```
mkdir ./mydb
```

Include and instantiate tingodb

```
var Engine = require('tingodb')();
var db = new Engine.Db('./mydb', {});
```

Insert data (row) into DB

```
var logCollection = db.collection('sensorlogs');
    var logEntry = { time: currTime,  first: first_val, second: second_val,
        third: third_val, fourth: fourth_val };
      logCollection.insert(logEntry, function(err, result) {
        if (err) throw err;
      });
```

DB data format of row

Insert into DB

Node and tingoDB -- example query (reading data from database)

Query without filter (**whole DB returned**)

```
db.collection('sensorlogs', function(err, collection) {
        if (err) throw err;
        collection.find().toArray(function(err, returned_data) {
             if (err) throw err;
             console.log("Database says: ", returned_data[1]);
```

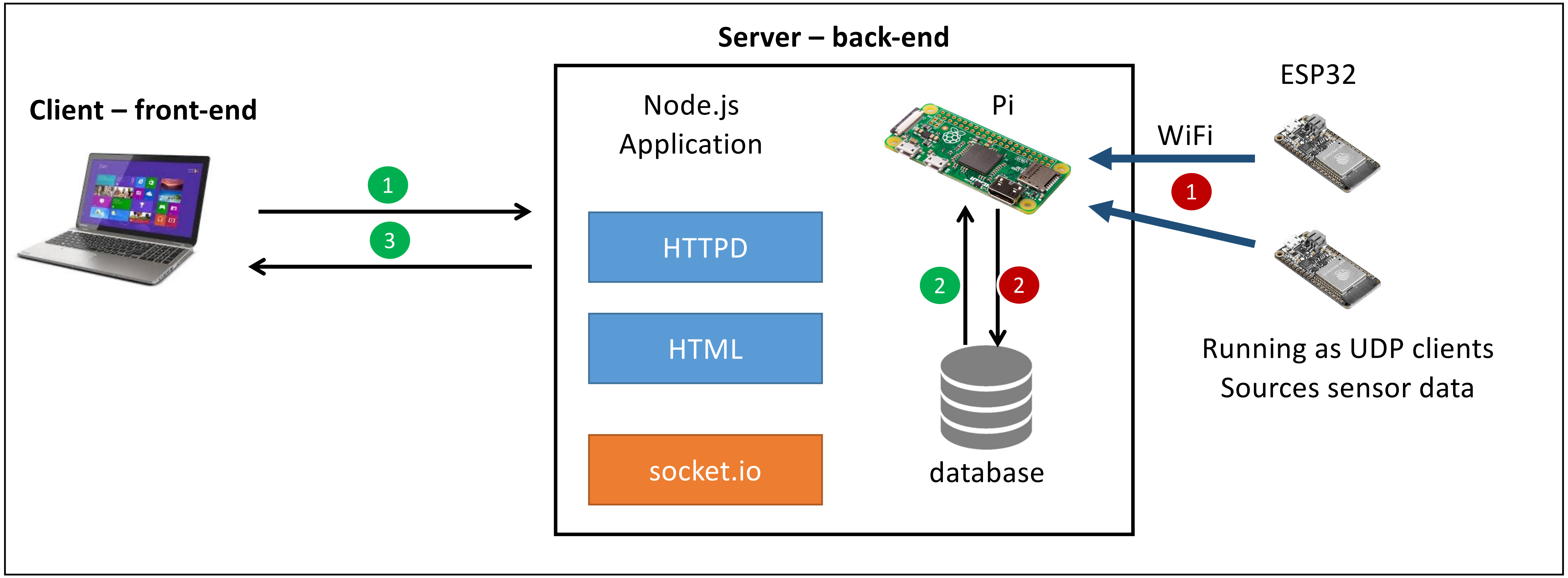
Filter here

Response from query (only showing first item)

```
{
  time: '2023-11-08T11:09:27-05:00',
  first: 1,
  second: 2,
  third: 3,
  fourth: 4,
  _id: ObjectID { id: 3 }
}
```

https://github.com/BU-EC444/04-Code-Examples/tree/main/tingo-sensor-data

Data flow in sensor system with database and client access



🟥 ESP32s generate sensor data and send to node app, node app writes data to database

🟩 Client requests data from node app, node app performs query to DB, responds with formatted results as HTML for display on client

https://github.com/BU-EC444/04-Code-Examples/tree/main/tingo-sensor-data

Database configuration with node.js server / quest

Approach – start with your laptop

- Node.js and your database (tingoDB)
- Create database matching your data table
- Create a query to show *n* rows of table
- Create html file to service your query – show *n* rows of table

Add a way to add a row

- Locally (create function to do so)
- Then use function when data arrives from an ESP32 message
- Sample code for adding a row for sensor data

Test and move to Pi

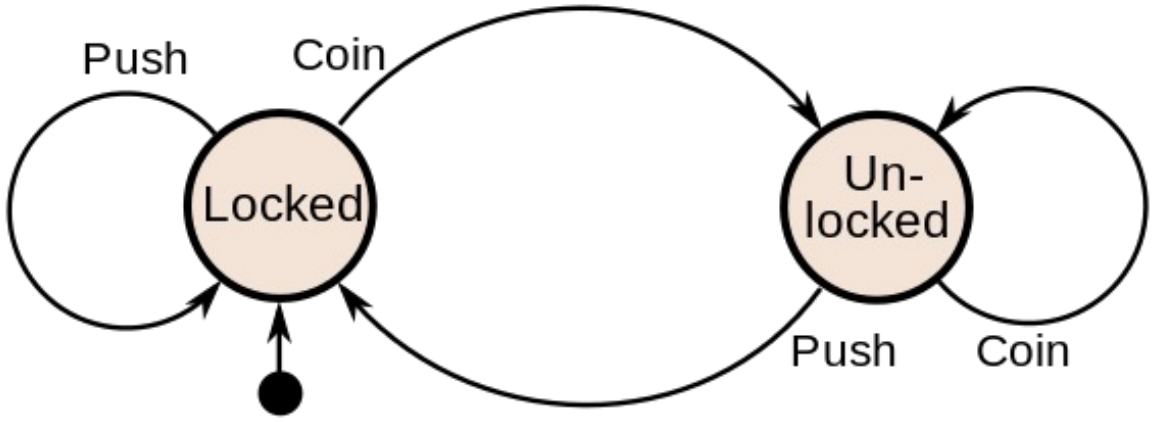
This is similar to the earlier quest except we save data to DB instead of csv file

Skill: State Models

Skill: State Models / Finite State Machines

Turnstile

FSM



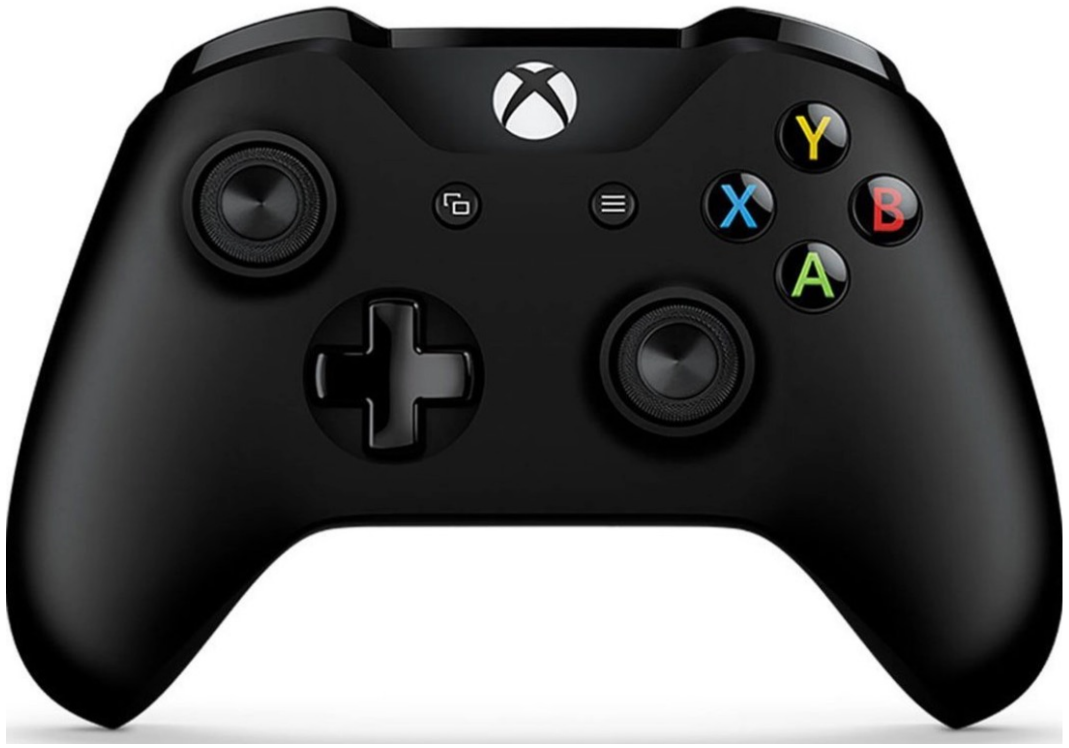
From Wiki

States (bubbles)
Transitions (lines)
Events (triggers on transitions)

State Table

Event	State	Next
Coin	Locked	Unlocked
Push	Locked	Locked
Coin	Unlocked	Unlocked
Push	Push	Locked

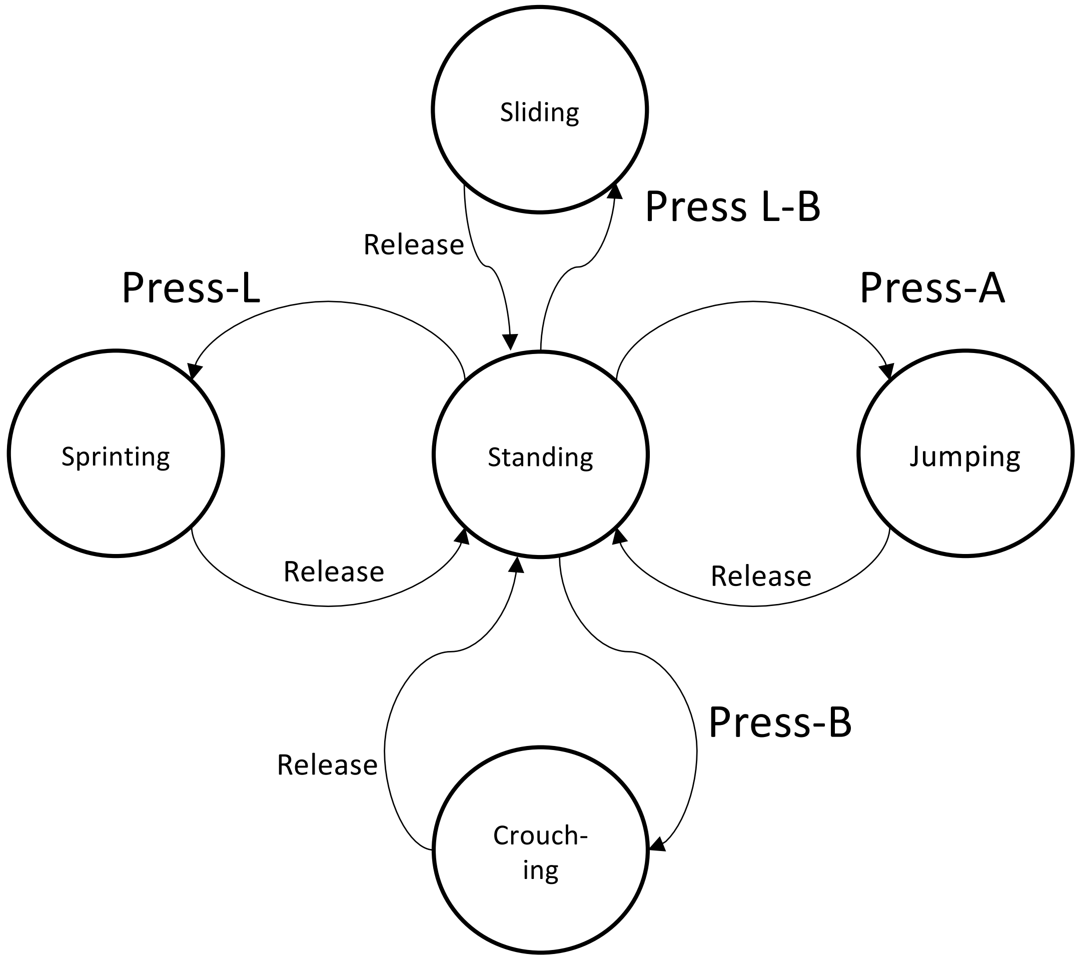
Example: game controller functions



States
- Standing
- Crouching
- Jumping
- Sprinting
- Sliding

Example: Model game control functions

States
- Standing
- Crouching
- Jumping
- Sprinting
- Sliding



State Table

Event	State	Next
L	Standing	Sprinting
A	Standing	Jumping
B	Standing	Crouching
L-B	Standing	Sliding
Release	{any}	Standing

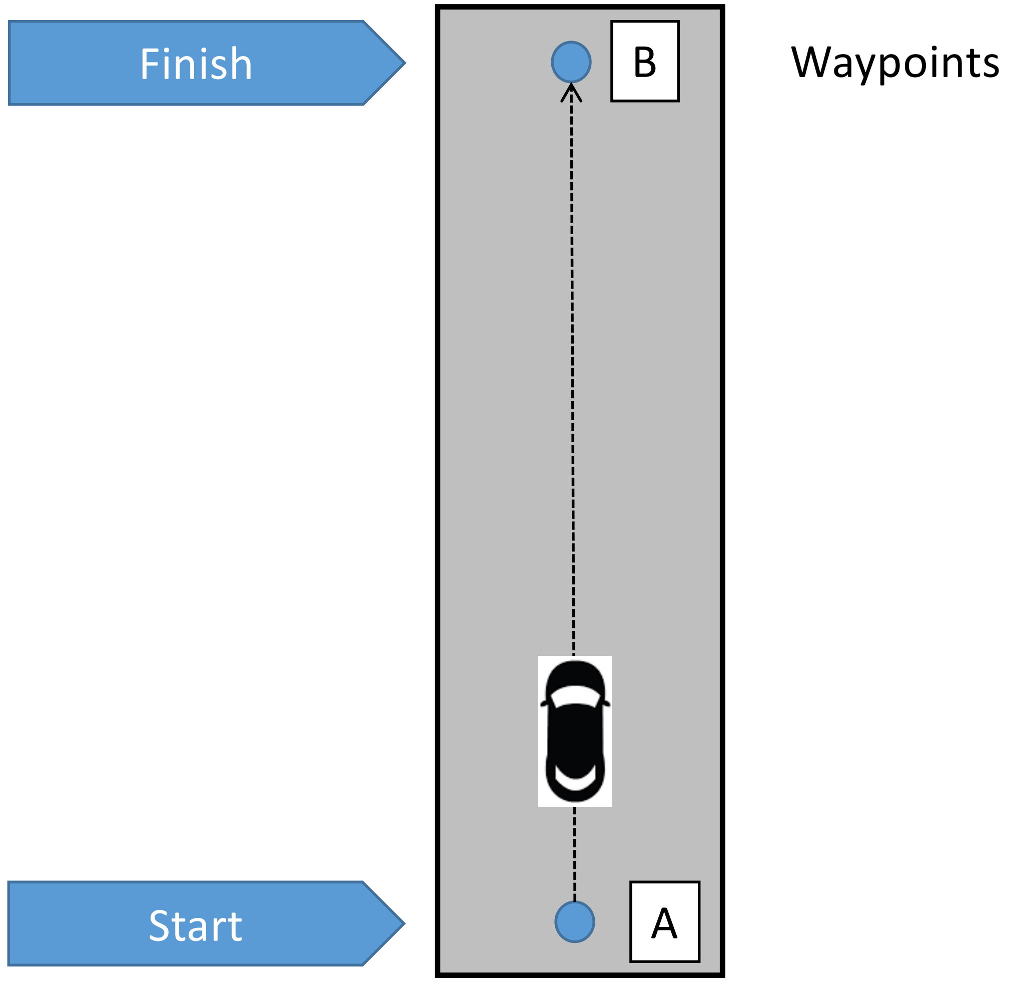
Driving as a state model

Autonomous driving simplified:

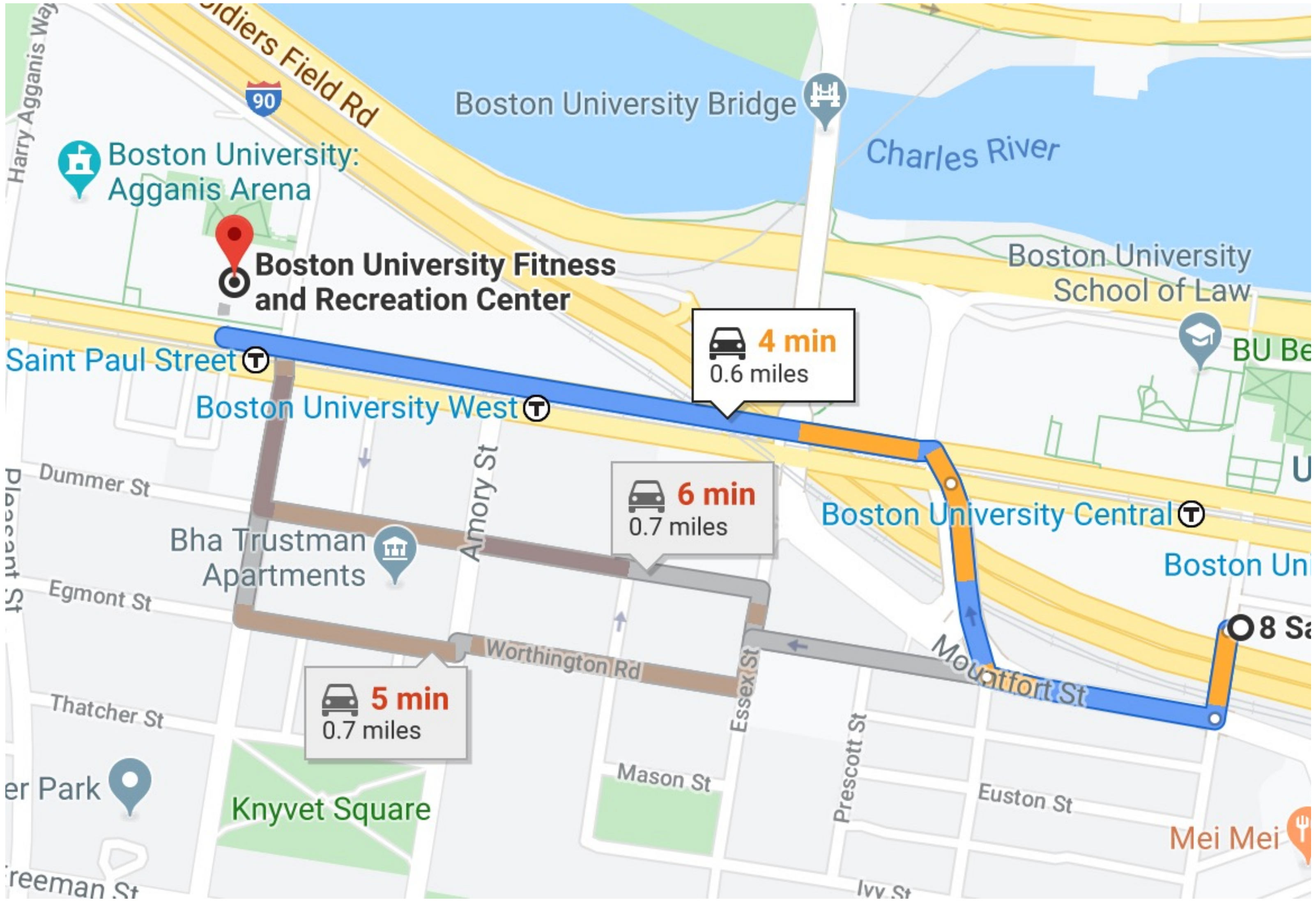
- Get from point A to point Z without crashing
- A→Z made of of many segments
- A→B→C→…→ Z
- Some segments curvy or irregular

- Key parts:
  - Drive the straight
  - Drive the corners
  - Not drive

States of Driving



A real example -- driving to Fitrec



8 St Marys St
Boston, MA 02215

Head south on St Marys St toward Mountfort St
230 ft

Turn right at the 1st cross street onto Mountfort St
0.1 mi

Turn right onto Carlton St
0.1 mi

Use the left 2 lanes to turn left onto Commonwealth Avenue
Destination will be on the right
0.4 mi

Boston University Fitness and Recreation Center
915 Commonwealth Avenue, Boston, MA 02215

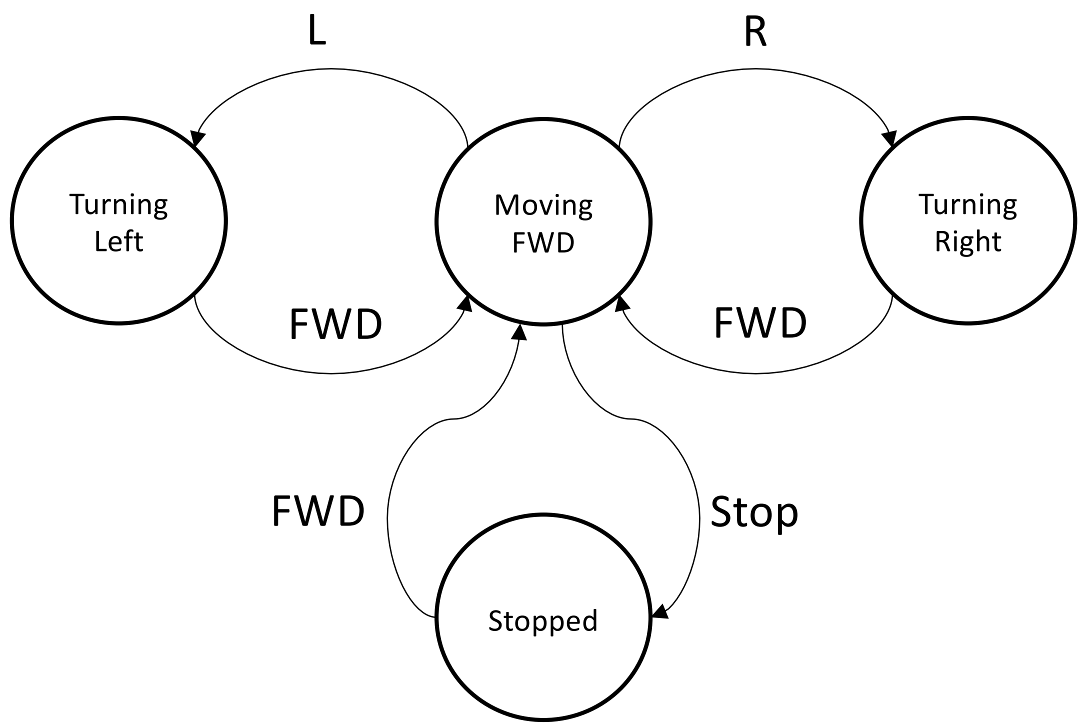
Example: Model driving as 4 states

From Google Maps

Path

• FWD:	St. Mary's
• R:	Mountfort
• R:	Carlton
• L:	Comm
• Stop:	Fitrec

FSM



Key takeaways:
- Model for driving is very simple and verifiable
- Model leads to compact implementation
- Can feed different paths to the model

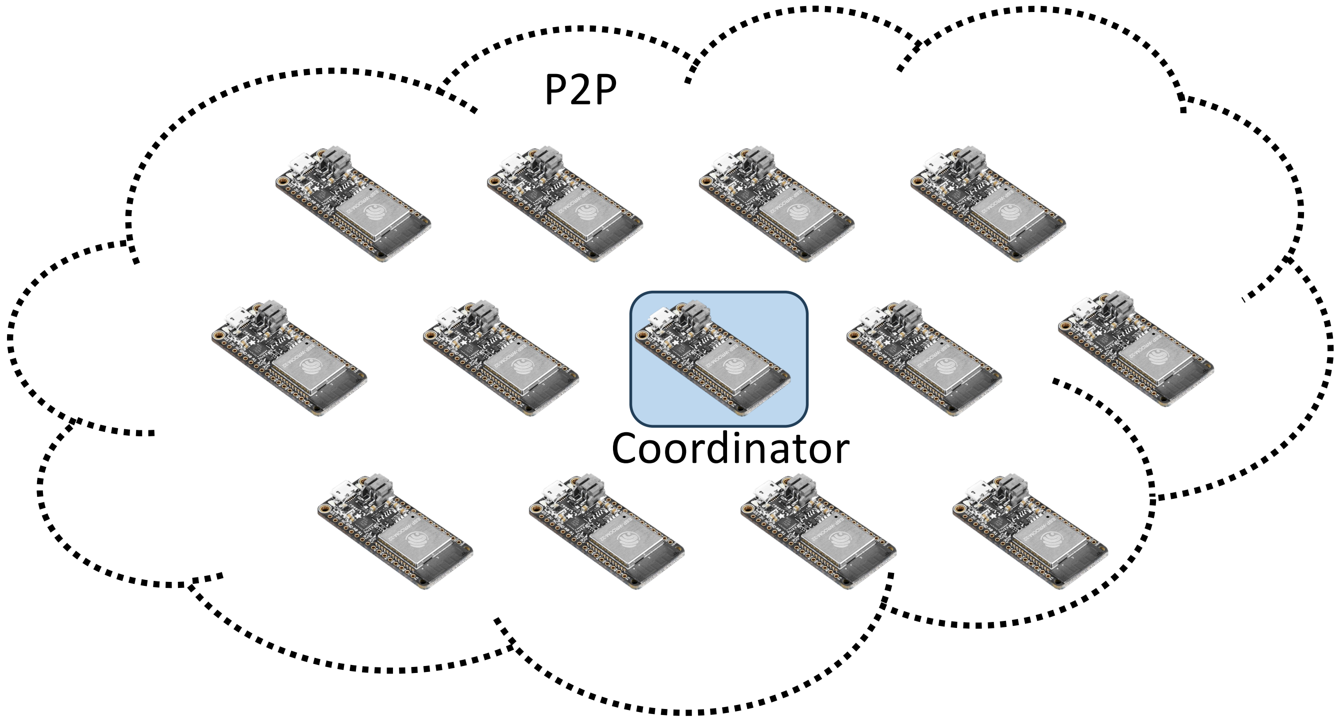
Skill: Using a service station to pump gas into a car

- Decide what are the states
- Decide what are the events and transitions between states
- Represent the FSM

- Draw up the FSM a gas pump interaction
- Create a matching state table
- Create matching C code based on the options in the design pattern

Skill: Leader (Coordinator) Election

Quest 5: How do we elect a coordinator in a distributed system?



Leader (Coordinator) Election

- Distributed computing
- Assign single process (node) as coordinator (leader)
- Needs to create a coordinator if absent
- Needs to be unique

See Wiki, “leader election”

Applications

- Wireless/mobile systems in which the coordination load will be rotated (e.g., energy)
  - Key distribution systems
  - Route coordination
  - Sensor/actuator control
- Systems in which optimize under a single coordinator or central—regional control
  - Car platoons
  - Medium access control
  - Data gathering (e.g., localization)
- Autonomous systems with many parts in which units can fail
  - Robot teams
  - Vehicle networks
  - Swarms
  - ATMs

Properties of our “Coordinator Election”

- Each node must decide if it is the coordinator or not; whilst ensuring there is only one coordinator
  - State of devices are “elected” or “not elected”
  - Once elected, stay elected (once not elected, stay not elected)
  - In every election, exactly one node becomes elected and the rest know that they are not elected

- Termination – finish in finite time
- Uniqueness – result in exactly one coordinator
- Agreement – everyone agrees who is the coordinator

See Wiki, “leader election”

How to do this? – common questions

Question	Technique to address
Who is in the universe?	Discovery
How detect changes in membership?	Hello messages/timeouts
Who is in charge?	Coordinator when elected
What if messages are lost?	Timeouts
What if nodes appear or disappear?	Maintenance
How to mitigate repeated flooding?	Optimization of algorithm

Typical approach to coordinator election – minimum ID

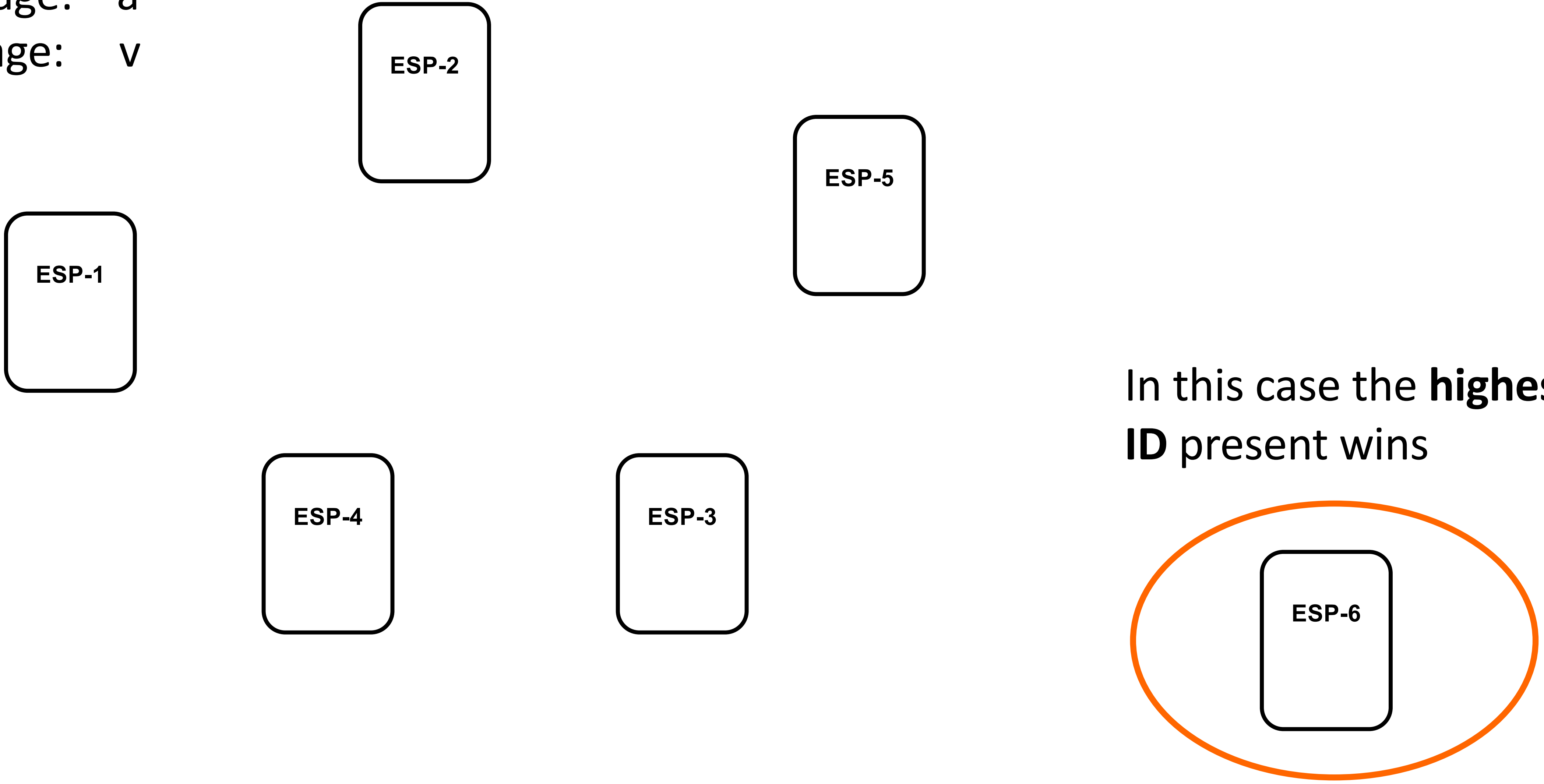
- **Discovery** – find all the participants – via each device broadcast (flooding)
- **Initiate election** – first one boots sends an election request
- **Respond** -- each device responds (broadcast) with their ID
- **Evaluate** -- each device receives the other IDs and can determine if they are the lowest
- **Confirm** -- The one with lowest ID signals (broadcast) that they are the coordinator

- Complexity comes from dealing with duplicate coordinators, intermittent connections, etc.
- **Code will be the same on each device**

- **We consider the Bully Algorithm**

Coordinator election with Bully algorithm

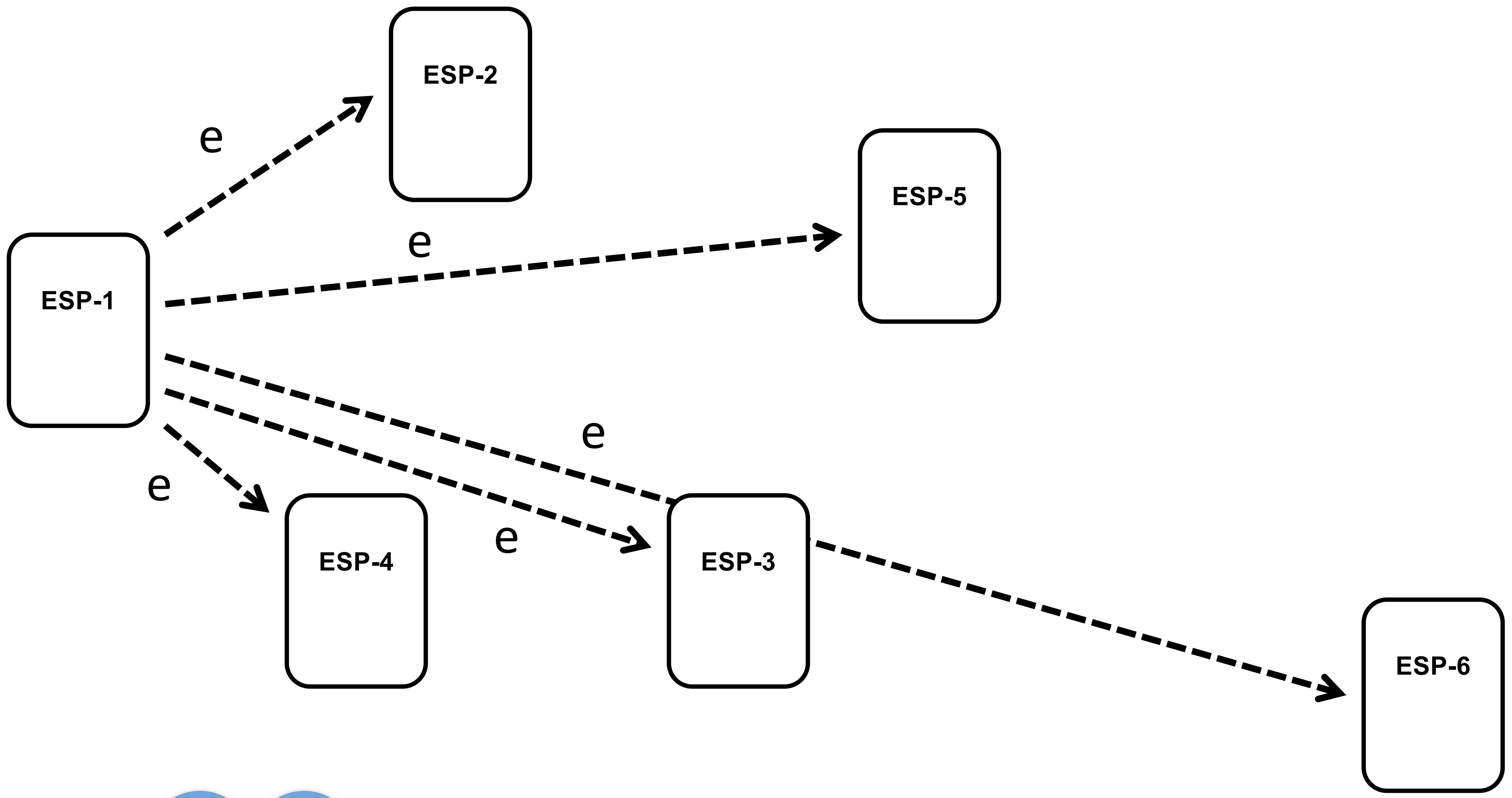
Election message:	e
Answer message:	a
Victory message:	v



In this case the **highest ID** present wins

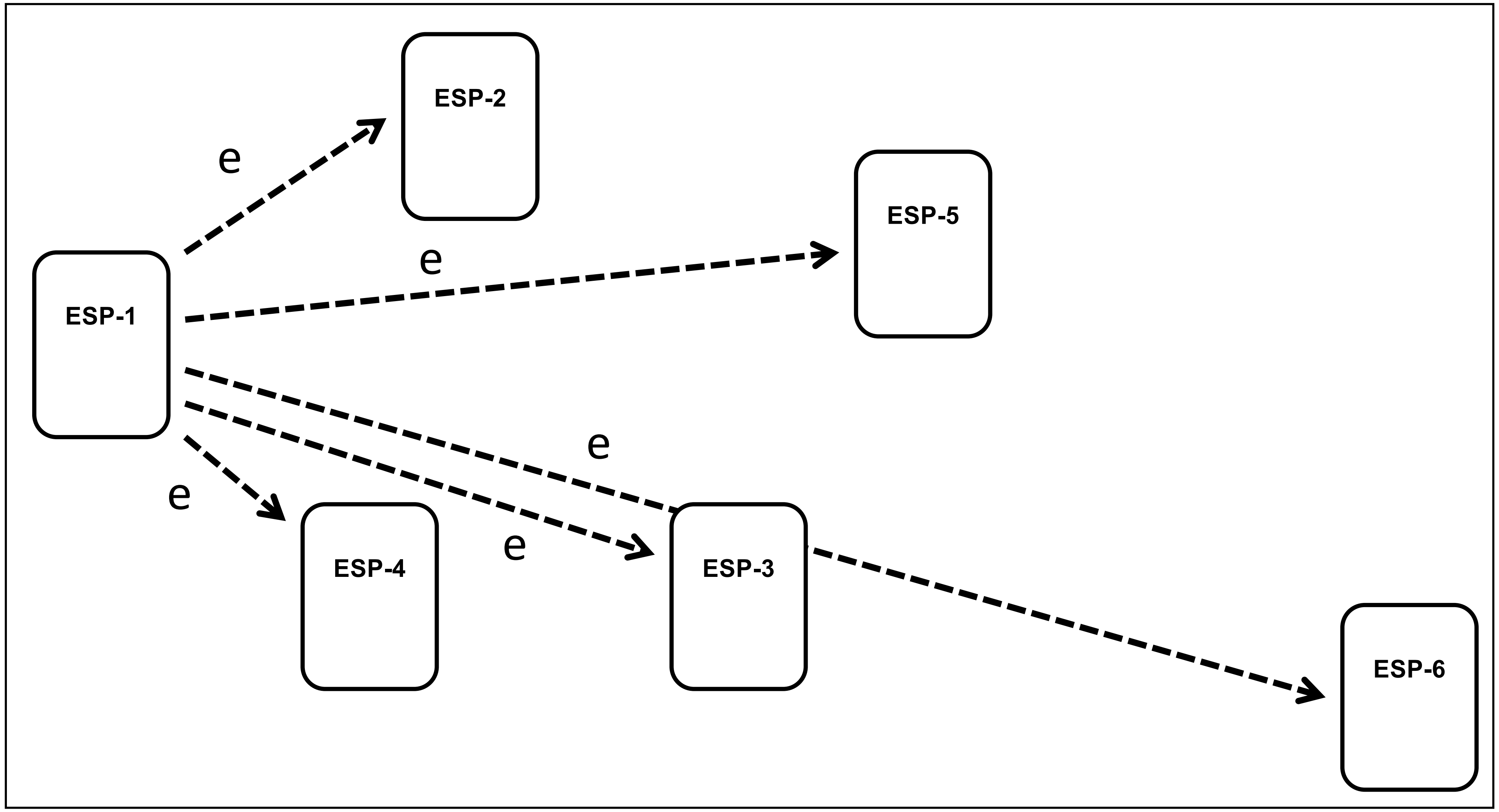
Bully algorithm

Shown from perspective
of a single node



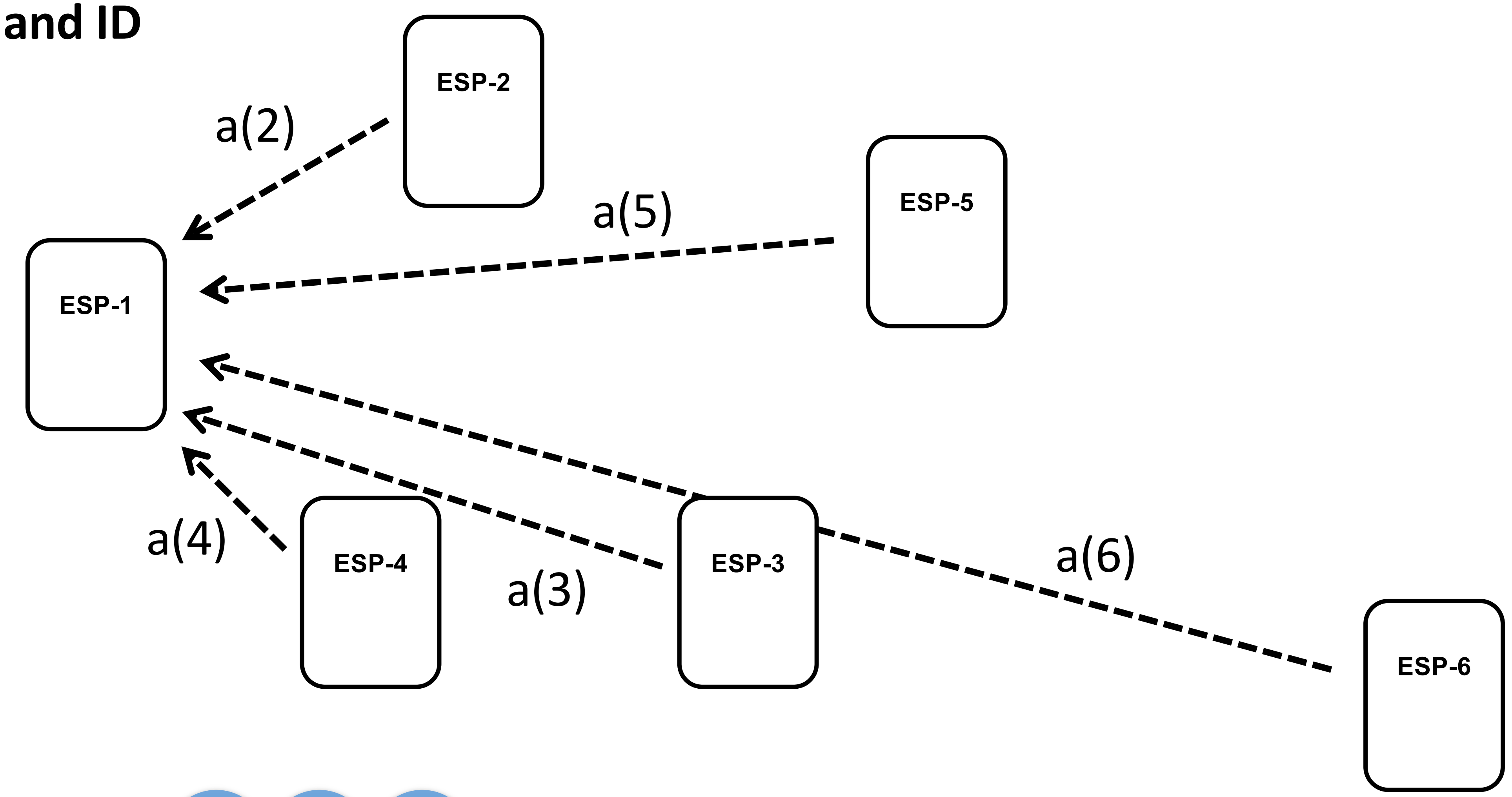
State 1 2

Bully algorithm



Bully algorithm

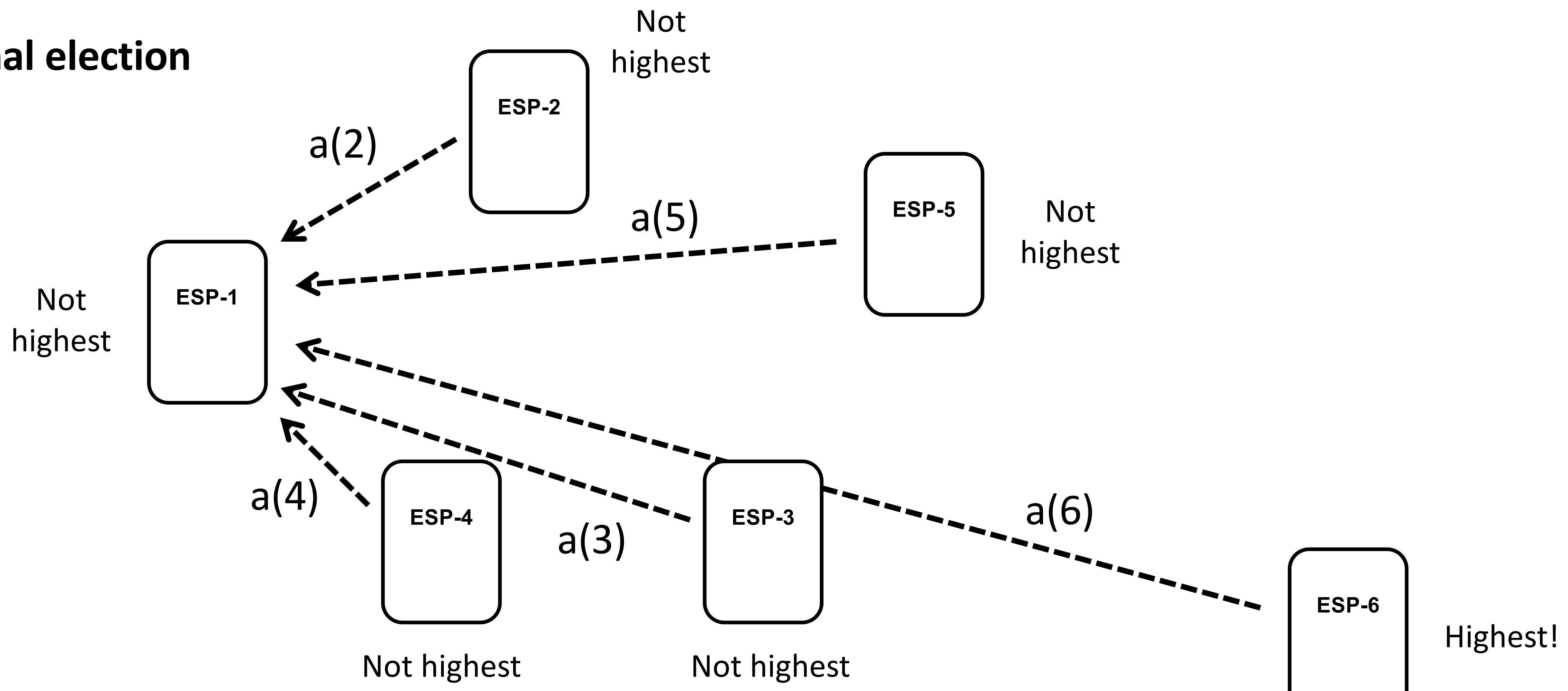
a – ‘answer’ and ID



State 1 2 3

Bully algorithm

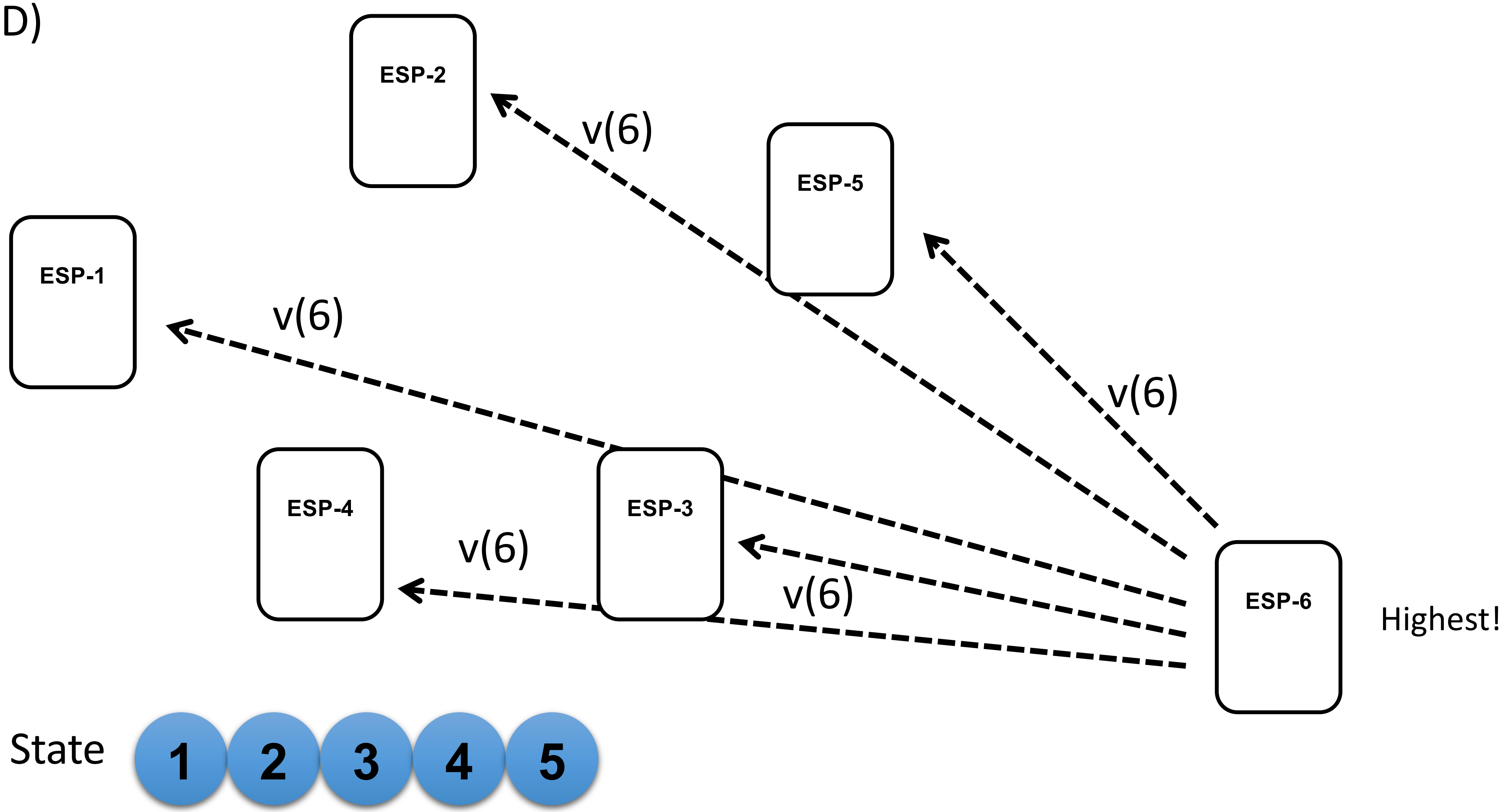
Normal election



State 1 2 3 4

Bully algorithm

v – ‘victory’ (ID)



Adaptation of Bully Algorithm for Implementation

Things to consider

1. You need a message format including type of message, and IDs communicated
2. You need to know who is in the system (universe of device IDs). This is IP:port addresses
3. You need to know who is connected (e.g., if they are detected on the network). This is IP:port addresses
4. What happens on a cold start? How does the election start?
5. How does the coordinator prevent restarting the election?
6. What happens if two separate networks form? Then they join?
7. Under what conditions does a device initiate a reelection?
8. What happens if one device fails during an election? (E.g., other devices wait for response that never arrives)

Some options

1. On cold start all devices **begin as coordinators**
   - Each device sends coordinator message to all other devices with own ID
   - All receiving devices decide if they have the lowest ID
   - Lowest ID remains coordinator; all others are not

2. On cold start all devices **begin as non-coordinators**
   - Devices timeout and then each send an election request
   - Each responds with ID; all devices pick lowest ID
   - Lowest ID responds as coordinator

What information should be in the message?

- Message type **(e, a, v, k), ID**

Use of Timeouts

Timeouts can mitigate problems with lost messages or devices disconnecting from the network

Possible timeouts implemented on each device:

- **Coordinator timeout** – the Coordinator failed to send me a keep-alive message (k)
- **Election timeout** – I have waited long enough to receive election messages so I will:
    1. Send a victory message and keep-alive messages (if lowest ID)

    OR

    2. Begin a new coordinator timeout period (if not lowest ID)

Modeling with states and events

- Model per device
  - Timeouts are events. (trigger transitions to new state)
  - Values of sent messages are events (trigger transitions to new state)
  - States are going to be something like
    - Leader
    - Election, but not a leader
    - Not a leader
- The overall system can be viewed as having a state too. But you are focused on the model of a **single device**

Summary details

- Message format (type, ID)
  a – answer
  e – election
  v – victory
  k – keep-alive
  ID – ID of sender
- Addressing
  ID, IP:port
  01, 192.168.1.10:3000
  02, 192.168.1.11: 3001
  03, 192.168.1.12:3002
  03,192.168.1.13:3003
- Timers
  Coordinator timer
  Election timer
  Keep-alive timer (Coordinator only)

Skill: Security Issues

Secure access and wireless

From August Lock

From wtop.com

From Bosch Mobility

How to configure without a keyboard?

How to ensure secure wireless communications?

Security issues in wireless networks (WiFi)

- Eavesdroppers / data interception / intrusion
  - Typically, this would be getting physical proximity to a network and tapping-in, could be a wired connection
  - Getting access behind a firewall, where protections are lower (e.g., a LAN vs a WAN)
- Spoilers (DOS)
  - Device does not conform to protocol and is disruptive (e.g., random backoffs)
  - Deliberate attack on protocols, for example, sending bad requests to a server
  - Often done by bots
- Ad hoc channels that bypass security
  - Listening to keystrokes or using modulate panel LEDs to decode data
- Endpoint attacks (pick on device)
  - Break into a fob, or thermostat, and interpret data by having the device, learning the credential
- Rogue access points (AP impersonation)
  - Free APs – log in and then they can sample your traffic
  - Also ”stingrays” in telephony
  - Impersonate an AP – trick you to logging in and exposing credentials
  - Sometimes called wireless phishing

Provisioning – no keyboard or monitor, how do you…

Without re-writing the firmware…

- Set up IP address (e.g., acquire address, and server knows address)?
- Configure device for a location (e.g., thermostat for room PHO 207)?
- Prevent someone else from configuring the device?
- Lock the access to the configuration by password? (Password needs to be exchanged)
- Prevent eavesdroppers (tapping the wireless exchange)?

Security features of ESP32

- Physical security
  - Encrypt flash, NVR
- WPA2,3 – using latest wireless protocols
- Lock-down ports – use only the minimum exposure
- TLS default integration (Transport Level Security/encryption)
- HTTPS – using TLS and authenticated endpoints with certs
- OTA programming and encrypted code distribution

BU IS&T – Endpoint Devices – Current rules (subset)

We can use the ESP32 connected to the campus network assuming:

- No confidential, sensitive, operational, or restricted information used
- Fewer than 10 devices
- Updates are done by physically reprogramming (re-flashing) the hardware device
- No remote or local logins are possible (the device is not running a local OS)

These are in the grey area:

- Recording authentication attempts would be difficult due to limited data storage (however, there is there is no "log-in" to the device).
- The device is not upgradable remotely — we don't plan to enable OTH (this makes it more secure, but if found to be compromised, must be physically unplugged)
- The device will not work with endpoint management software unless we also use it with a physical gateway (which is how it has been used in the past)

Skill Assignment

Suppose you are assigned the task of driving your car/robot via remote control over the internet.

1. **Sketch the overall flow** of information to drive a car with remote control over the Internet.
2. **Identify weaknesses** in your overall system (client, local network, internet, server, node.js, ESP32) with respect to security compromise.
3. **List at least five ways** a malicious actor can attack your specific system. Be very specific
4. **Describe a way to mitigate each attack**
5. **Write up your answers and report** -- please include graphics as necessary to support your response