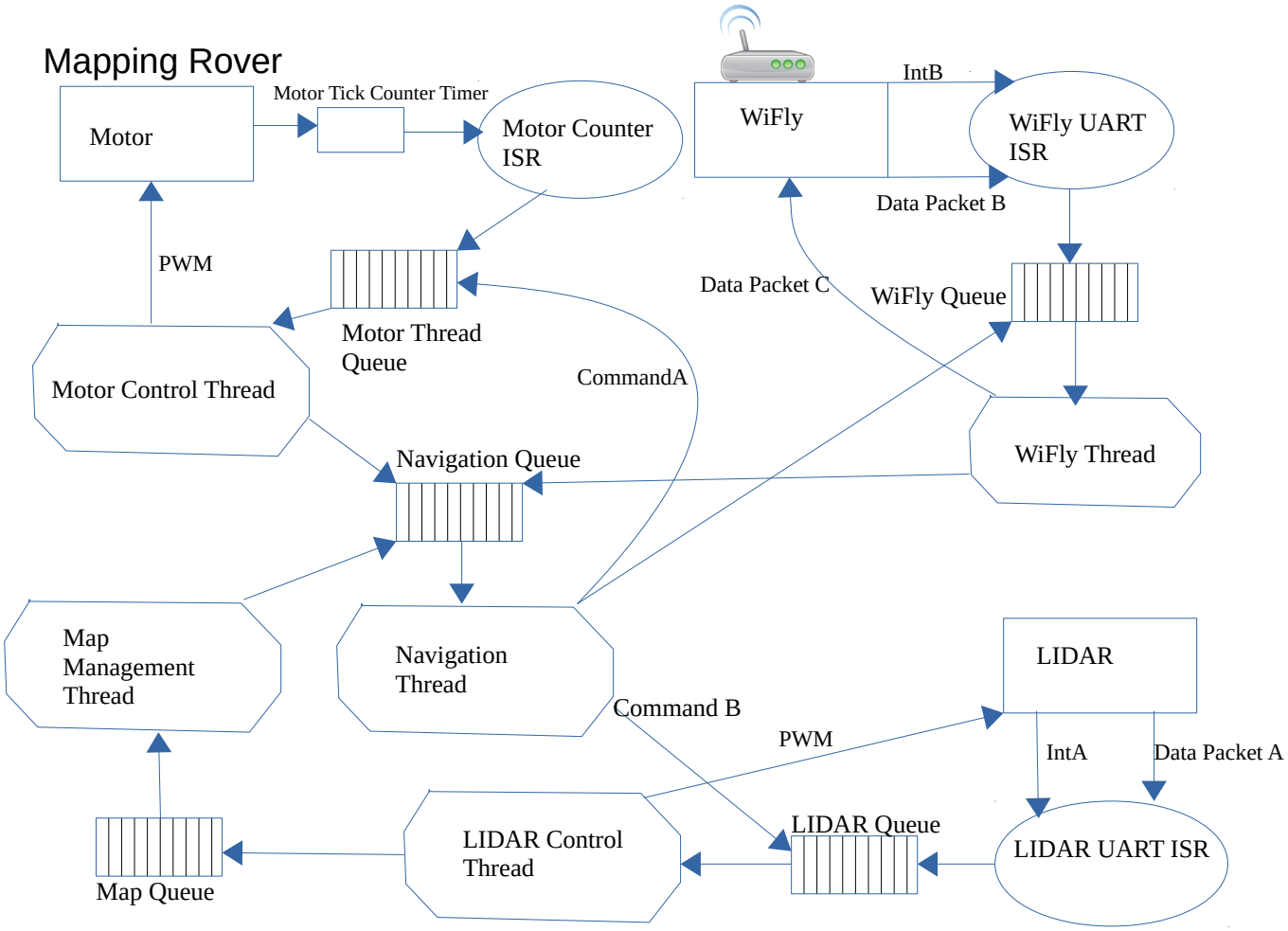
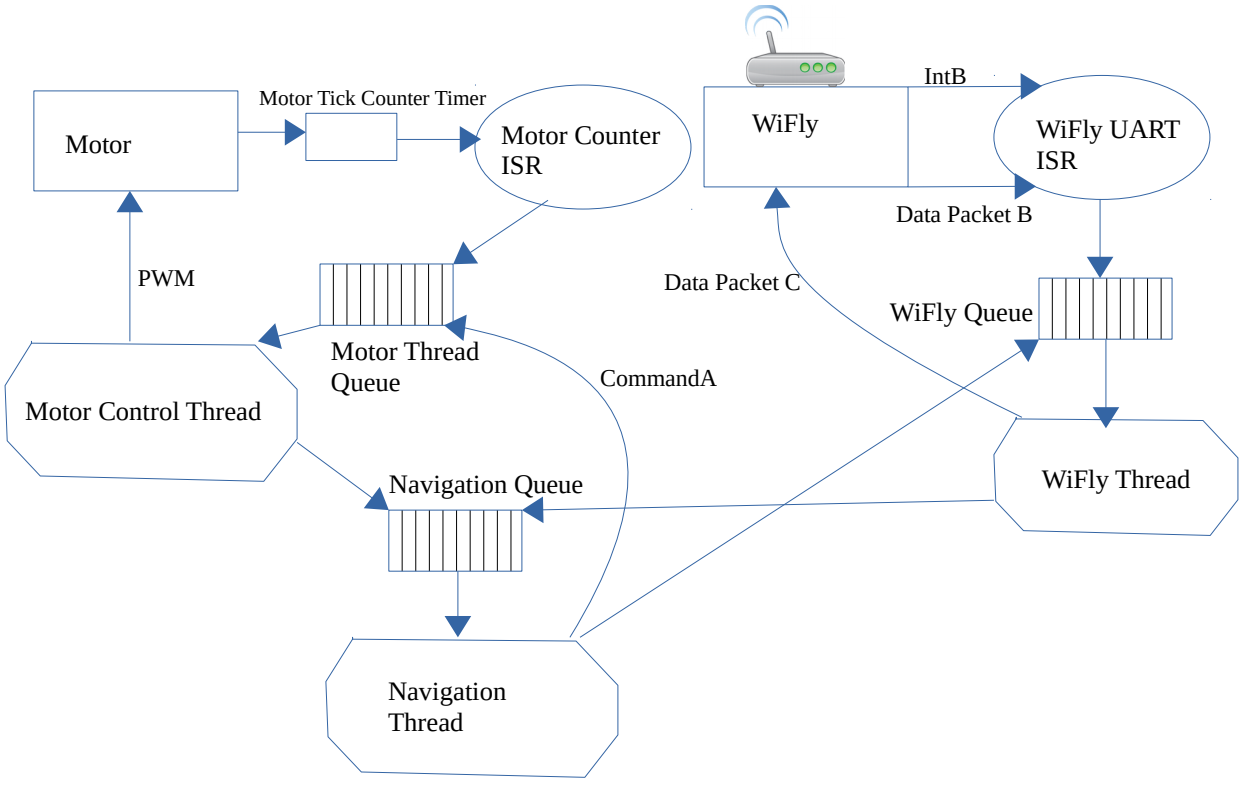
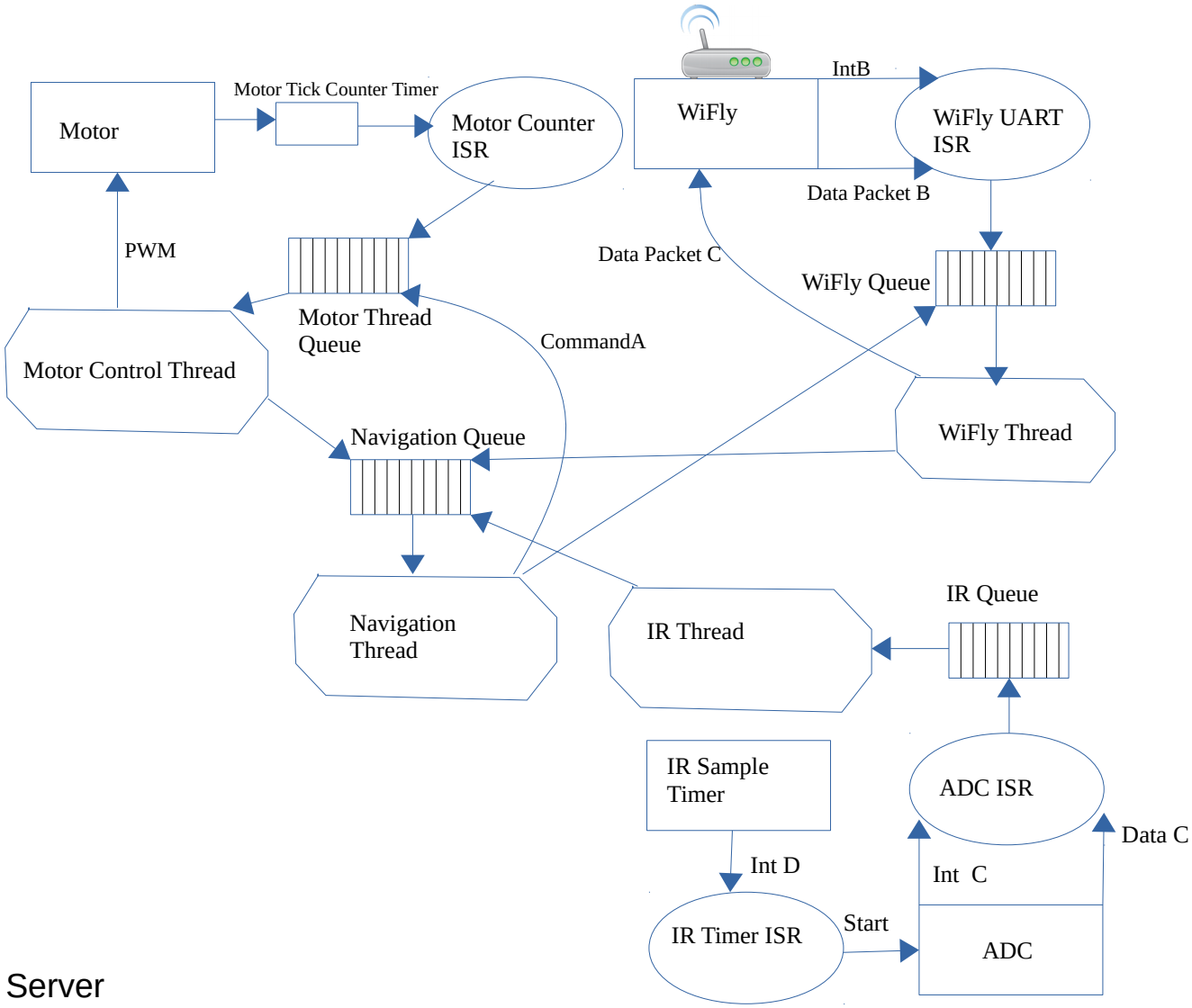
Mapping Rover



Hiding Rover



Seeking Rover

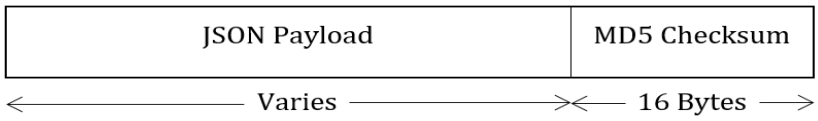


Server

External Communications

Messages over WiFi will be sent at a variable rate. From intermittent up to 10 kB/s

Message Layer



JSON Payload Format

{“cmd”: “store”, “type”:“sensor_value”, “data”:12345.678}

Description

Shared Threads – Threads that are the same across all rovers:

- Motor Control – This thread accepts commands and data from the Motor Thread Queue. The commands come from the Navigation thread via Command A and consist of a direction and distance in which to move in degrees and cm. The Motor Control Thread then executes the commands by measuring the distance traveled via the data coming in and by outputting a PWM signal to activate the motor. It then sends to measured distance traveled in cm to the navigation thread.
- WiFly – This thread accepts commands from the navigation thread about connecting to the server. If there is to be an outgoing message, this thread encapsulates it in Data Packet C and sends it to the WiFly Module. If waiting for data to come in, it waits for Data Packet B to arrive on the WiFly Queue.

Unique Threads – Threads that are unique to a particular rover:

- LIDAR Control – This thread controls the LIDAR sensor and receives and parses data from it. It receives control commands (Command B) and Data Packet A from the LIDAR Queue. The sampling rate is controled via a PWM input to the LIDAR module. It then parses Data Packet A consisting of a degree offset from straight and a distance in cm from the LIDAR module and sends the data to the Map Queue.
- Map Managment – This thread takes in data from the Map Queue and constructs a 2D map of the room the rover is in. It does this by starting the rover at (0,0) and using the LIDAR data to plot points on the x,y plane. This is done by using the offset degrees and distance in cm to plot the points. When the map is updated, the updates are sent to the Navigation Queue for use by the Navigation Thread.
- IR Thread – This thread receives IR reflective data in from the IR Queue and sends that data to the Seeker Rover's Navigation Queue.
- Mapping Rover's Navigation – This thread is in charge of determining how the room is to be mapped as well as determining when the room is fully mapped. It first waits for the start command to come from the WiFly Thread by sending the appropriate wait command to the WiFly Queue. Then it activates the LIDAR Module via Command B. Once the initial data is processed by the Map Management Thread, an initial path is determined by avoiding obstacles immediately in front of the rover. Command A is then used to tell the motor to follow the initial path in small increments of a few cm. Once the data from the Motor Control Thread about the actual distance traveled comes in, the thread will move it place in the map and send another Command A. This procedure repeats until an obstacle is encountered 15-30cm away. From here, the thread will determine where to turn to avoid obstacles by prioritizing paths that lead towards the less explored areas of the map. It continues to use the Command A, wait for distance traveled, move position on map (ADP) loop to execute this. After a few thousand loop iterations, the thread will start checking to see if the map has changed recently as part of the navigation loop. If the map has not changed, then the thread determines that mapping is complete. It will then stop the motor via Command A, stop the LIDAR via Command B, and send the map data to the server via the WiFly Queue. Additionally, the Navigation Thread can accept a stop command from the server via the WiFly in order to stop mapping and send the map data.
- Hiding Rover's Navigation – This thread directs the hiding rover to a specified location on the map. On reset, the thread waits for the start command to come from the server via the WiFly.

Once this is received it requests the map data from the server and waits for a response. Once the map data has been received, the thread waits for a destination to arrive from the server via the WiFly: (x,y) @ degrees offset. Once the destination arrives, the thread checks that the rover can fit in the specified location with a few cm clearance on all sides to allow for turning the rover. If the location is invalid, the thread will notify the server via the WiFly Queue. If the location is valid, the proper path is set by first assuming that the rover is starting at (0,0) @ 0 degrees offset then by moving in a straight line towards the destination. It continues this path until an obstacle is met and then navigates around it. This process continues via the ADP loop until the rover is at its location. Once the rover arrives, the thread sends a completion message to the server via the WiFly Queue.

- Seeker Rover's Navigation – This thread functions similarly to the Hiding Rover's Navigation Thread. However, an additional step of confirming that the Hiding Rover has been located via the IR sensor data is added. Once the data coming from the IR Sensor Thread confirms that the hiding rover is in range of the Seeker rover, the thread will then send a completion message to the server via the WiFly Queue.

Shared Devices – Devices that are the same for all of the rovers:

- Motor – The motor controls the movement of the rover as well as sends movement data as output via quadrature encoding. Motor control is given via PWM on the input pins. The movement data is sent out on two separate channels; one for each tread of the rover.
  - Along with the motor is the Motor Counter Timer. Because of the quadrature encoding, the distance traveled is reported in ticks. This timer is used to count those ticks and send an interrupt when a sufficiently high number of ticks has been counted. The timer rolling over represents that the motor has gone some standard unit of distance. It outputs Int E.
- WiFly – The WiFly is a UART connected device that allows for network communication via WiFi. It takes in commands for connecting to the network and sending messages via UART. It also sends received messages to the connected UART module (Data Packet B).

Unique Devices – Devices that are unique to a particular rover:

- LIDAR – This device uses a LIDAR sensor to measure the straight-line distance in front of the sensor to the nearest obstacle within a 2 cm resolution. The range is up to 40m. The LIDAR sensor is mounted on a 360 degree motor which can rotate at a variable speed. This rotation is controlled via PWM on the motor control input pins. The data is formatted as packets containing both distance in cm and the offset from 0 degrees (Data Packet A). This data is sent via UART.
- IR Sensor – This device uses an IR emitter to detect if the surface in front of it is reflective of IR radiation. This data is analog so an ADC is required to sample the data. It outputs Int C and Data Packet C.
- IR Timer – This timer sets the sample rate for the IR Sensor. It outputs Int D.

Shared ISRs – ISRs that are the same for all of the rovers:

- Motor Counter – This ISR is used to handle when the motor counter timer reports that the motor has gone a certain distance via Int E. The ISR sends a simple message representing the distance traveled to the Motor Thread Queue for processing.
- WiFly UART – This ISR handles IntB and received Data Packet B coming in via the WiFly. It

unpacks the data into a useful format and forwards it to the WiFly Queue for further processing.

Unique ISRs – ISRs that are unique to a particular rover:

- LIDAR UART – This ISR handles Int A and the LIDAR sensor packets (Data Packet A) coming in from the LIDAR sensor. It simply unpacks them and forwards them to the LIDAR Queue for processing.
- IR Timer ISR – This ISR handles Int D when the IR Timer rolls over. It activates the ADC for conversion and returns.
- IR ADC ISR – This ISR handles Int C when the ADC conversion is complete after being activated by the IR Timer ISR. It reads and unpacks the ADC data and forwards it the the IR Queue for processing.

Shared Queues – Queues that are the same for all of the rovers:

- WiFly – This queue accepts commands from the other threads as well as data received from the WiFly ISR.
- Motor – This queue accepts Command A from the navigation thread as well as distance traveled data from the Motor Timer Counter ISR.
- Navigation – This queue accepts distance traveled data from the Motor Thread, Data Packet B from the WiFly Thread, as well as data from all connected sensors. In the case of the Mapping Rover, this is the LIDAR and Map Management Thread. For the Seeker Rover this is the IR Thread.

Unique Queues – Queues that are unique to a particular rover:

- LIDAR – This queue forwards Data Packet A data from the LIDAR ISR to the LIDAR Queue.
- Map Management – This queue forwards map data points from the LIDAR Thread to the Map Management Thread.
- IR Queue – This queue forwards IR Sensor reading data from the IR ADC ISR to the IR Thread.

Internal Message Format:

- All internal queues will have the same message format:
  - 86 bit struct with three fields:
    - 32 bit ‘command’ field. This is a unique command/response identifier
    - 32 bit ‘type’ field. This specifies the data types stored in the last field
    - 32 bit variable ‘value’ field. This field will hold any value that can be represented with 32 bits, from byte to float.