gecko

Wireless E-Stop

2024.08.01 | Philip Kuhle | Final Presentation

Problem

Problem

Xbox controller connectivity is unreliable.

Guiding Principle

Every failure mode should result in the robot losing power.

Wireless Controller Hardware

Radio - Frequency Selection

- Current: 2.4 GHz Bluetooth
- Wireless E-Stop: 900 MHz
- Why use a lower frequency?
    - Generally experience less attenuation
- Drawback to lower frequency?
    - Lower throughput, but it doesn't matter too much for this application

Radio - Module Selection

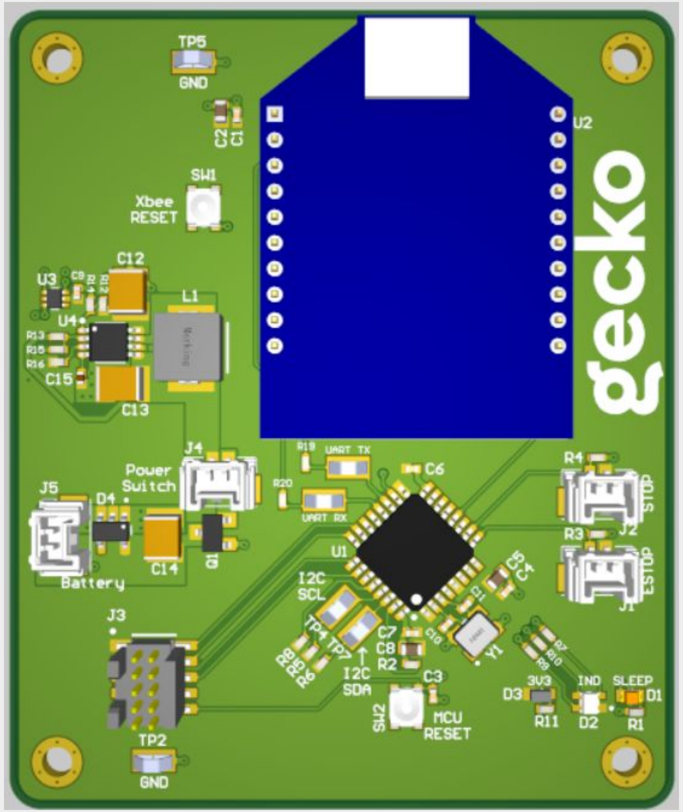
- Chose Digi's XBee
- Already had some in stock
- Prototyping benefits
  - Digi XCTU
- Simplicity
  - UART device
  - RF stuff handled inside the chip
- Currently configured at 24 dBm output power

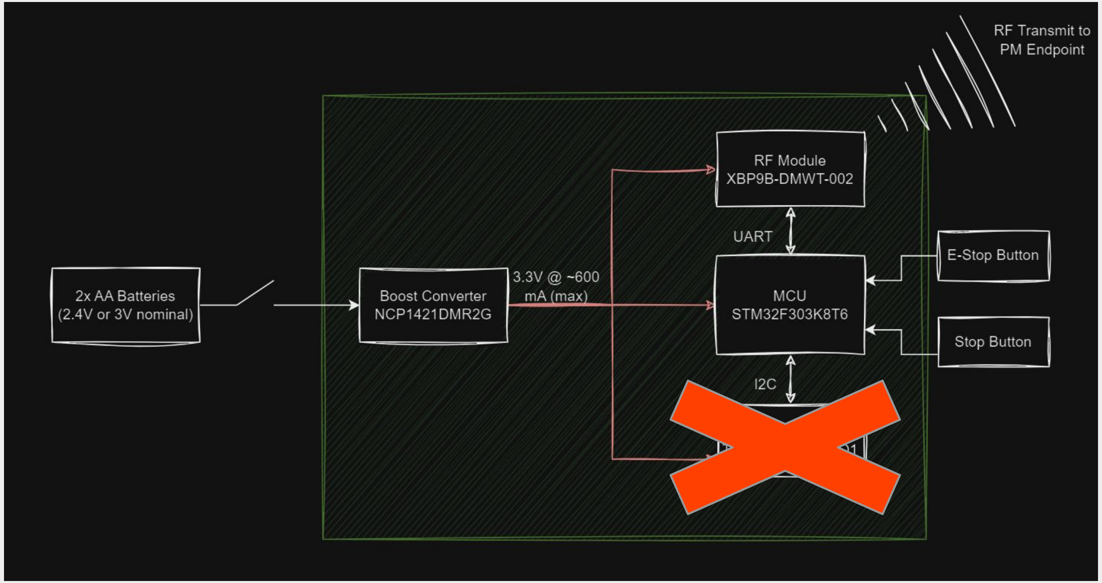
Controller Batteries

- Goal: Make sure operators are not changing batteries frequently
  - Target battery life was ~60 hours = 5 days x 12 hour shifts
- Worst case battery life estimate: 100+ hours for 2x disposable AA batteries
  - Can change depending on packets sent and how frequently they are sent
  - Assumptions
    - TX = RX = 0.6 ms
    - TX and ACK pair every 0.5 seconds
    - Remaining time, Xbee is sleeping
- Regulator designed such that controller works using disposable or rechargeable AA batteries
  - Regulator also has a low battery detect pin

Buttons

- Design needed to have two buttons on it
    - One button for E-stop (stop robot by cutting power)
    - One button for stopping robot without cutting power

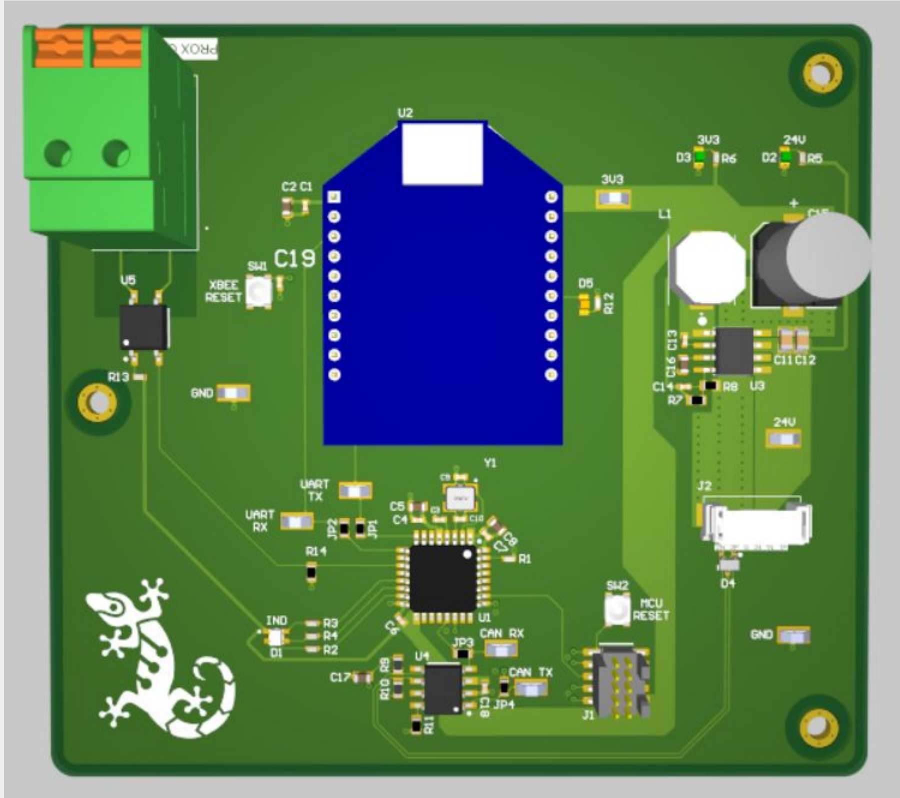

TP5
GND
Xbee
RESET
SW1
U2
gecko
C12
U3
L1
U4
C15
C13
J4
Power
Switch
J5
D4
Battery
C14
Q1
UART TX
UART RX
C6
U1
I2C
SCL
I2C
SDA
R4
STOP
R3
J2
ESTOP
J1
J3
C7
C8
R2
C3
Y1
MCU
RESET
SW2
3V3
IND
SLEEP
D3
D1
R11
D2
R1
TP2
GND

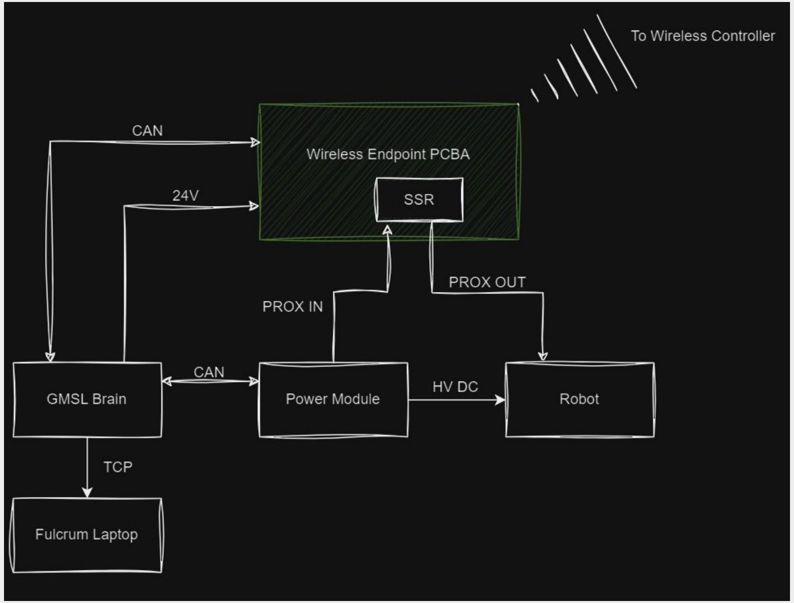

RF Transmit to
PM Endpoint
RF Module
XBP9B-DMWT-002
UART
E-Stop Button
2x AA Batteries
(2.4V or 3V nominal)
Boost Converter
NCP1421DMR2G
3.3V @ ~600
mA (max)
MCU
STM32F303K8T6
Stop Button
I2C

Wireless Endpoint Hardware

Switching Off Robot Power

- Three options
  - Interrupt AC
    - Not viable since PCBA is powered by the PM brain
  - Interrupt DC to robot
    - Best solution, in my opinion, since it is most direct
    - Implementation is difficult right now
    - Would be most viable on a new revision of the PM
  - Interrupt prox
    - Easiest way to prove E-stop concept
    - Not the most direct path to cutting power since it relies on the prox HW and FW working on top of the wireless stuff working


U2
C2 C1
C19
SW1
XBEE RESET
U5
R13
GND
3V3
D3
R6
24V
D2
R5
L1
D5
R12
C13
C16
C14
R7
R8
U3
C11 C12
Y1
UART TX
UART RX
JP2
JP1
C5
C4
C7
C8
R1
R14
IND
R3
R4
R2
D1
U1
C6
CAN RX
JP3
U4
C17
R10
R9
CAN TX
JP4
R11
SW2
MCU RESET
J1
J2
D4
GND


To Wireless Controller
CAN
Wireless Endpoint PCBA
24V
SSR
PROX OUT
PROX IN
CAN
GMSL Brain
Power Module
HV DC
Robot
TCP
Fulcrum Laptop

Firmware

Overview

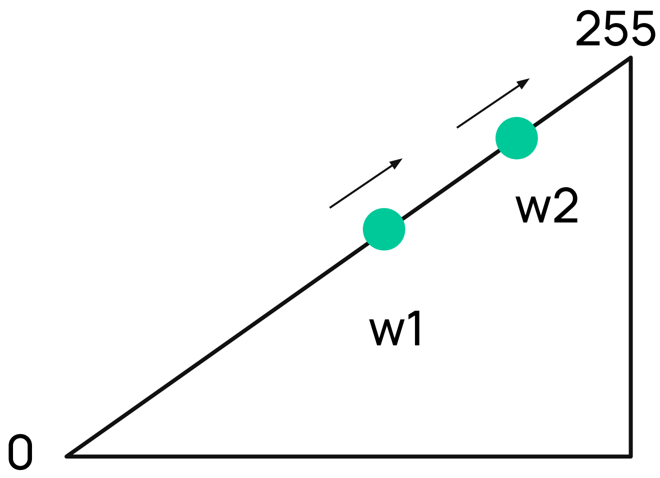
- Controller sends different message to endpoint depending on which buttons are or are not pressed
- Controller can monitor different conditions of itself such as temperature and battery life and communicate those to the endpoint
- Both controller and endpoint can figure out if they are disconnected from each other
  - Mostly done through use of custom watchdog
- Controller state machine
- Endpoint state machine
- Didn’t get the chance to implement state machine

Watchdog - General Idea

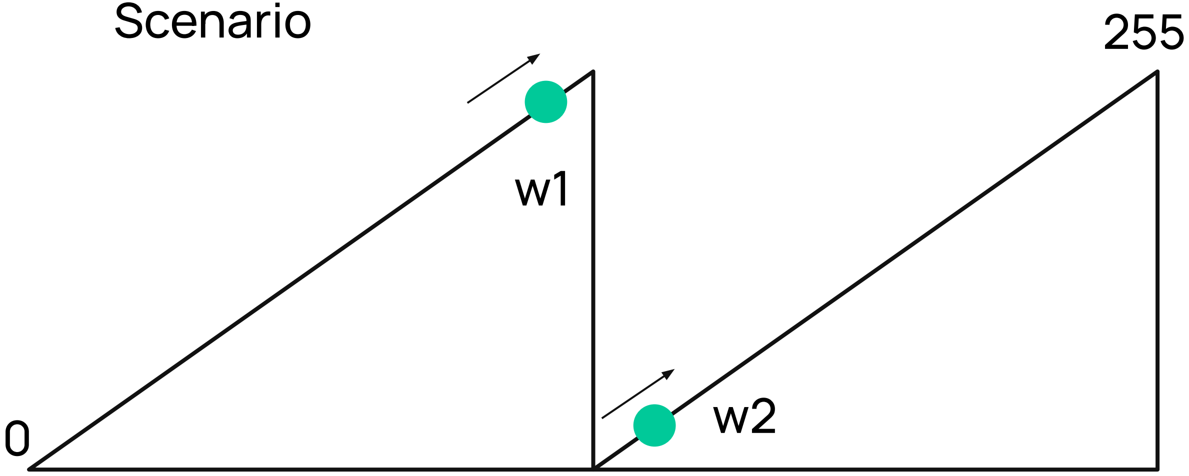
- Controller sends endpoint a counter value
- Endpoint receives controller value and compares it to an expected value
    - If the controller value is within a window centered around the endpoint's expected value, the transmission is accepted
        - The expected value is incremented and an ACK is sent to the controller
        - When the controller receives the ACK, it increments its counter
    - If the controller value is outside the window of acceptable values, the transmission is rejected
        - Controller is deemed to be disconnected and robot power is switched off

Watchdog - Initial Idea

Easy Scenario

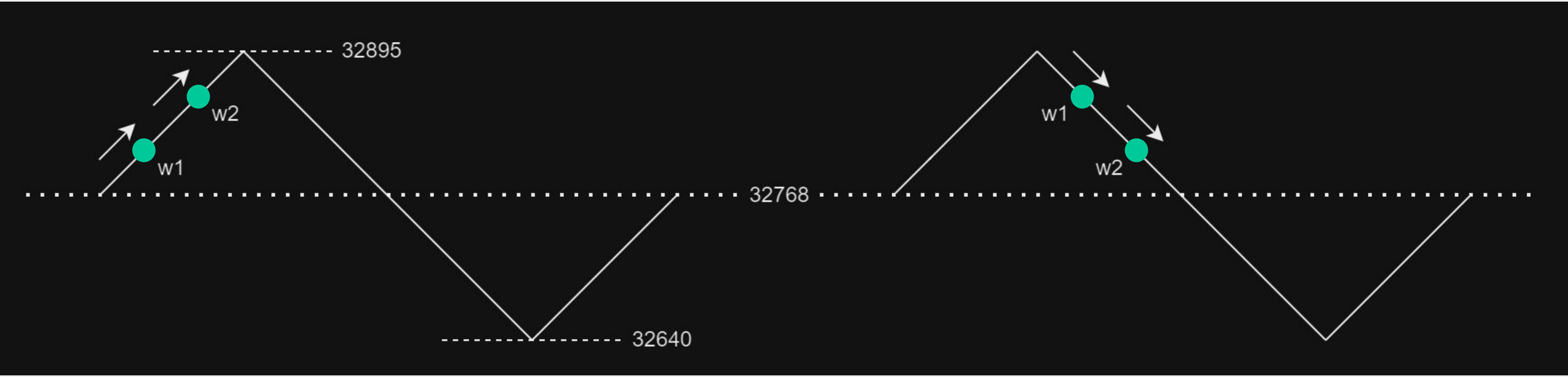


Not-So-Easy Scenario

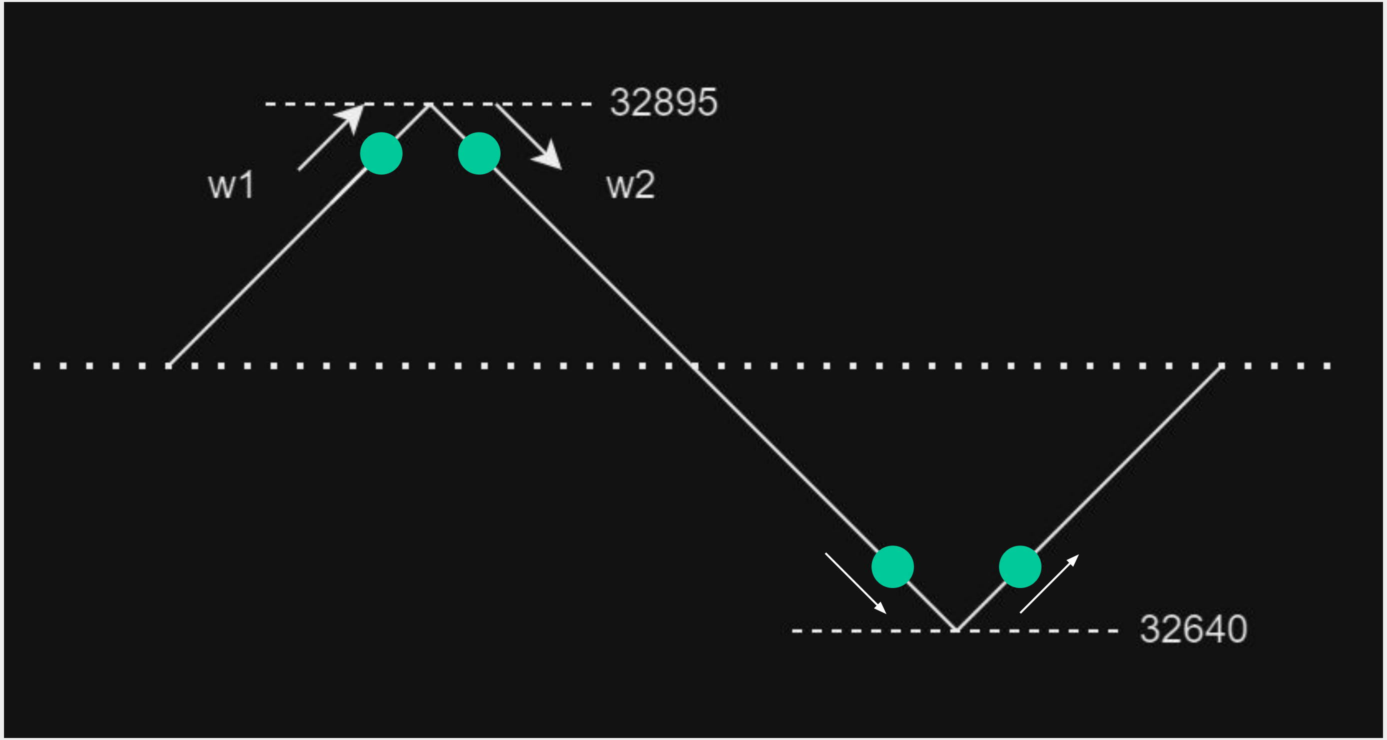


Watchdog - New Idea

- Two 16-bit counters that operate in 8-bit range
- (UINT16_MAX / 2) + INT8_MIN ≤ (UINT16_MAX / 2) ≤ (UINT16_MAX / 2) + INT8_MAX



Case 1



Case 2

Lessons Learned

Lessons Learned

- Better project planning and requirement setting
    - Was asking myself questions in the middle of my internship that I thought I had already addressed. This means that the foundation of my project wasn't solid enough and could have been prepared better.
- Focus on the novel part of the problem
    - Got too caught up in the nitty gritty hardware details at the beginning and lost valuable time near the end.

Special Thanks