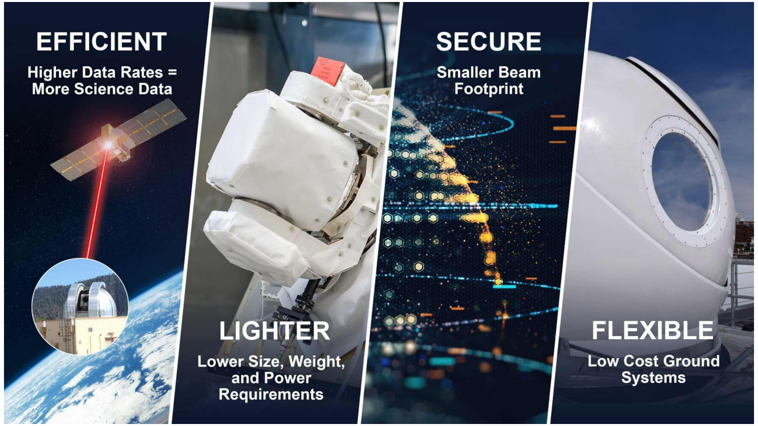
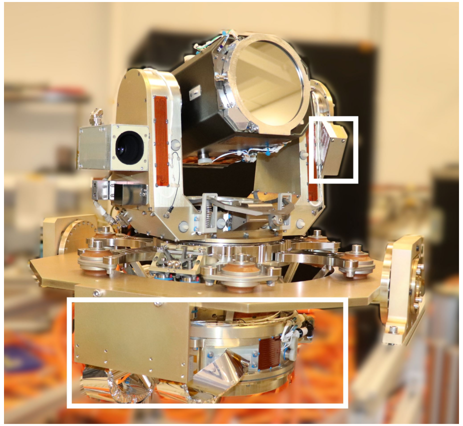
Team 07 Demo Video

Noah, David, Lezly, Ephraim

Executive Summary

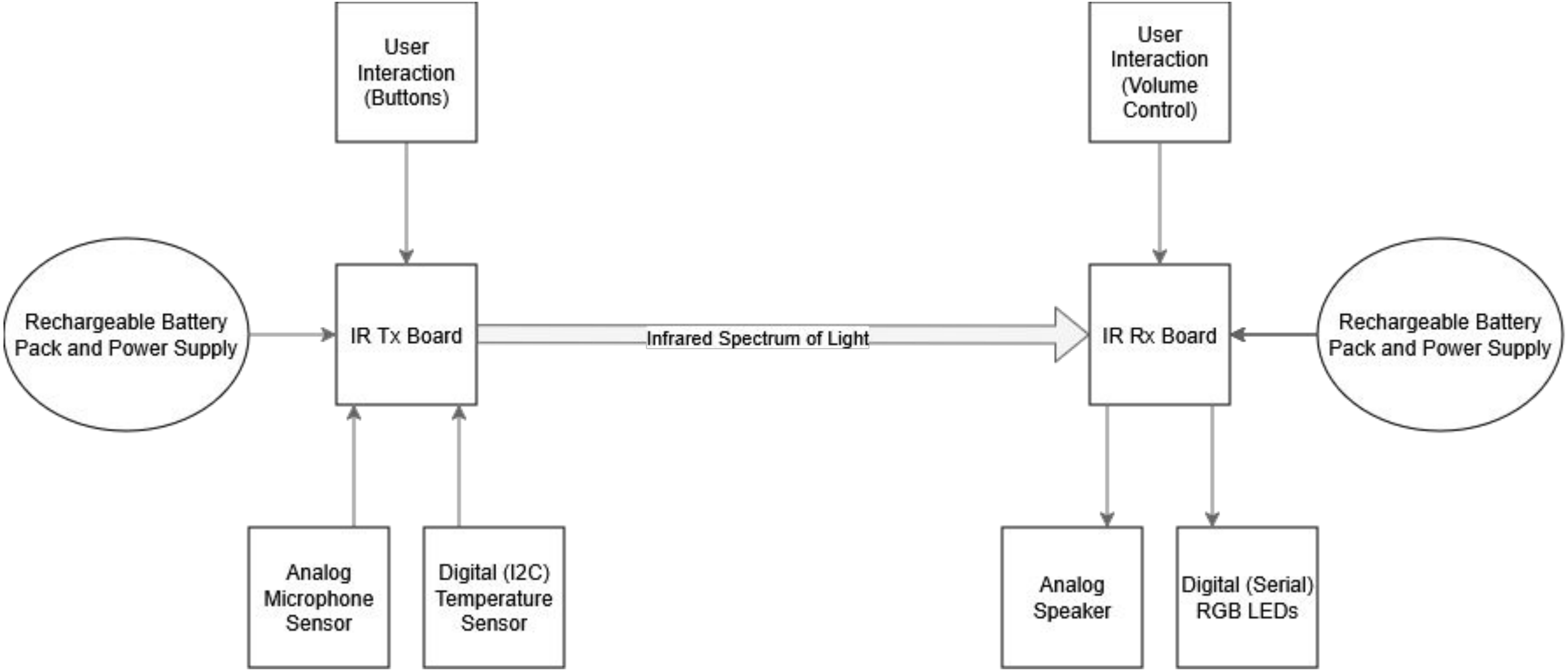
- Understand principles and challenges of light based communication
- Use generalized hardware to send encoded digital data (and digital representations of analog) over infrared spectrum.
- Implement a custom mix of software to send data efficiently and reliably
- Do this on a budget and with low power hardware





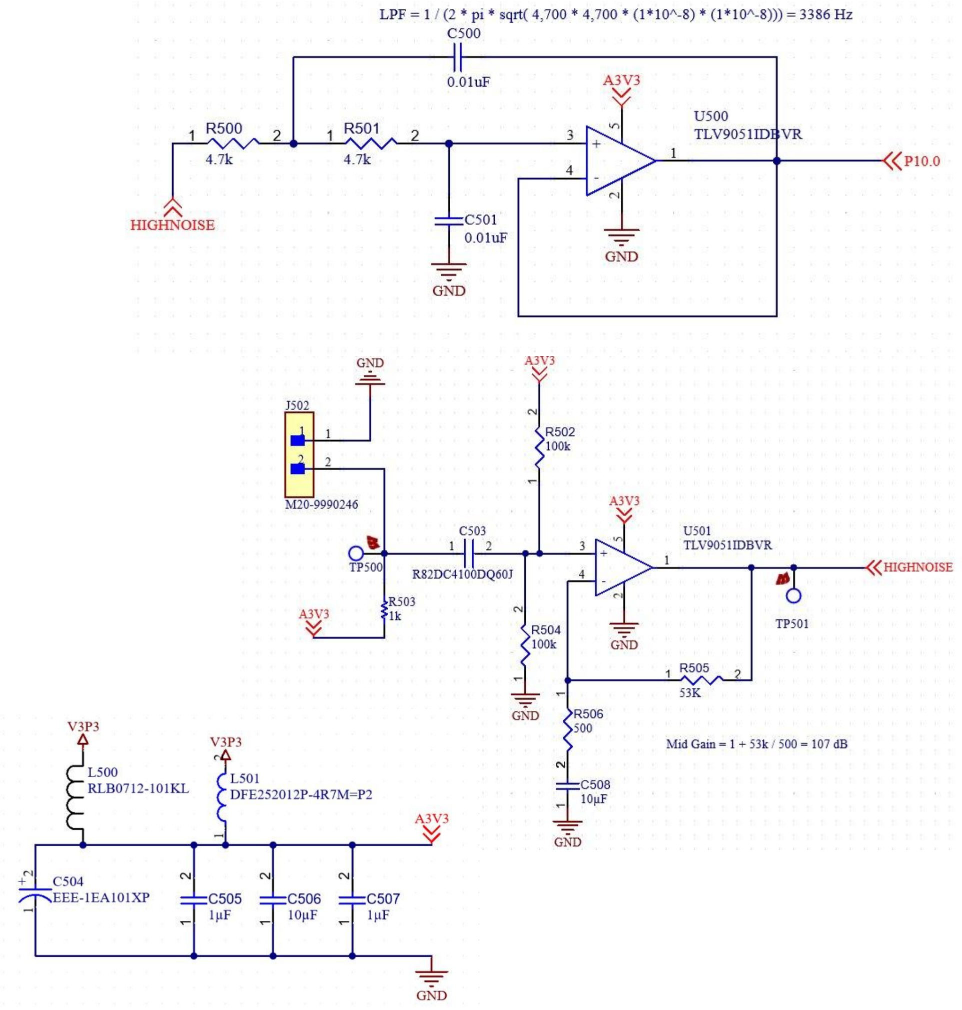
"…serving as a high-speed data pipeline between the astronauts on Orion and mission control on Earth. O2O will beam information via lasers at up to **260 megabits per second (Mbps)** to ground optical stations"

Executive Summary Diagram



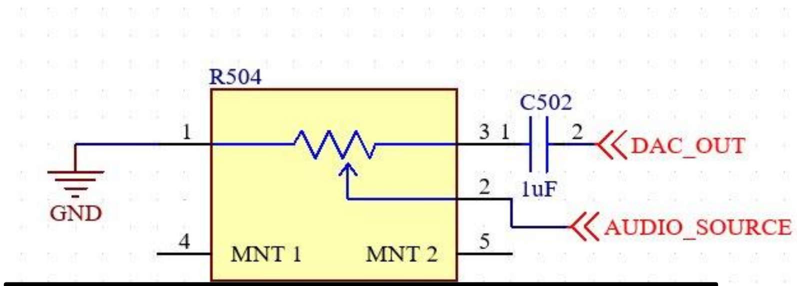
Hardware Summary - Mic

- CMA-4544PF-W microphone is a raw component with –44dB
- First, amplify with non inverting, HPF stage to get up to a workable amplitude
- Then, run through unity gain LPF in order cut off frequencies above ~3.3 Khz, to prevent aliasing effects
- Use power filtering for analog amps

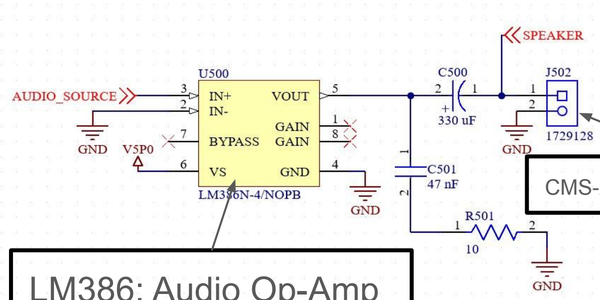


Hardware Summary - Speaker

- First, amplitude of audio sample was sent to DAC, between 0 and 3.3 V
- Voltage was then sent through capacitor, to center -1.65-1.65 V around 0 V
- Next, signal sent through potentiometer, to increase or decrease volume/voltage
- Then, signal sent through audio amplifier, for 26 dB voltage gain
- Finally, speaker outputs sound based on difference between Vs and Gnd.



Potentiometer (used for volume control)



LM386: Audio Op-Amp

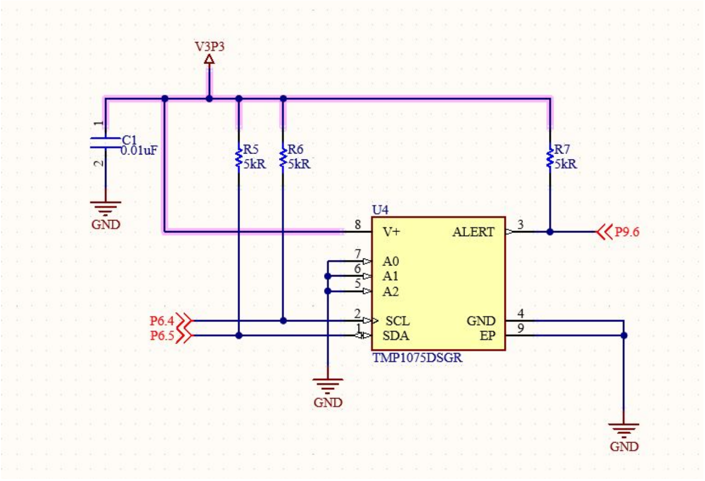
CMS-28588N-L152: Speaker

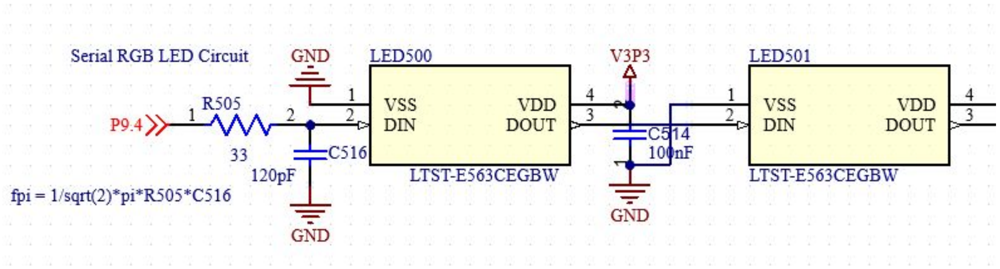
Software Summary - Mic / Speaker

- ADC sampling rates
  - Samples around 65 Khz, but average down to 8 Khz in order to reduce load on FreeRTOS
  - Runs continuously, but has a capture window that gets activated by command
  - Timer counts down in background to stop capture window
- Storing in queue vs array
  - Queue: frequent context switches, gets stored in FreeRTOS heap
  - Static Array: simple, easy to add to, gets stored in .bss, static RAM
- Codecs
  - Sending all the stored data (160 KB in 10 seconds) over IR is super slow
  - Compressing it with a codec allows faster transmission
  - Compiled our own codec2 implementation, an open source audio codec
- MSB Decimation
  - Codec proved too expensive for PSoC 6 silicon
  - We chose to mask the lower half of each 16 bit sample for faster transmission

Hardware Summary - Temp Sensor & RGB LEDs

- Temperature sensor TMP1075 has a designated register for holding the temperature value and could read by I2C protocols.
- The serial RGB LED are cascaded to receive color signal in order.
- There is low pass filter to filter out the high frequency noise in the input signal.





Software Summary - Temp / RGB LEDs

- Temperature Sensor

  Temperature value was read in from the register of a TMP1075

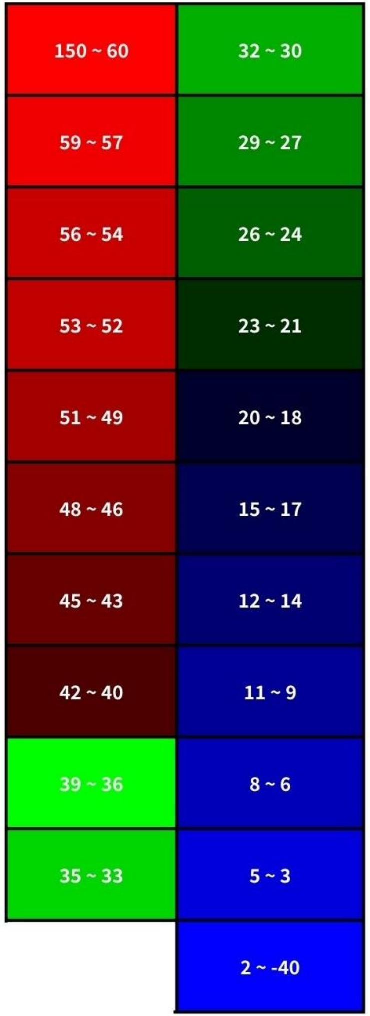
  The value is then mapped to a corresponding temperature interval

- RGB LEDs

  The LEDs read in a 24-bit serial signal, each color is 8 bits

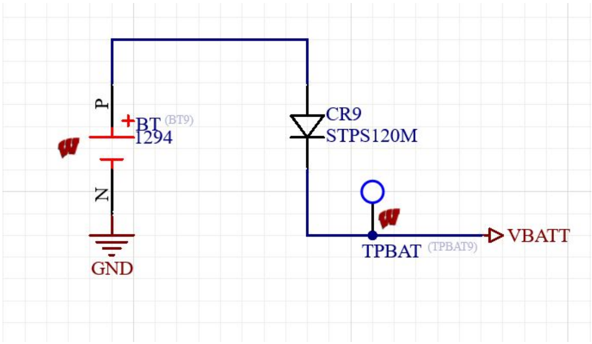
  The corresponding color display on all of the LEDs

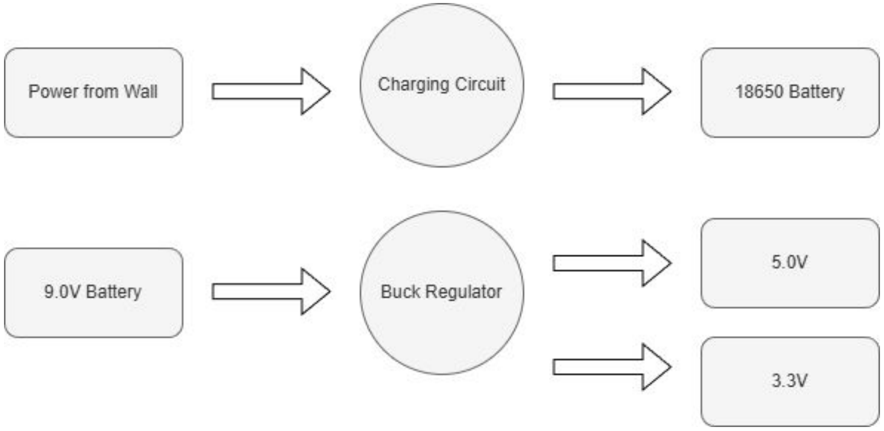
  Also serve as loading indicator, while 1 light indicate 1000 sample was received.



Hardware Summary - Battery

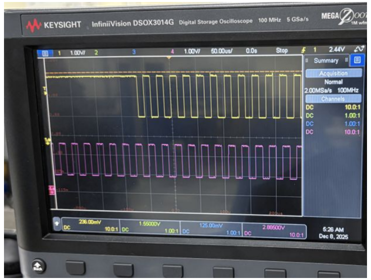
- Buck regulator is used to reduce the 9V battery to 5V and 3.3V
- Use a MCP73831T-2ATI/OT to charge an 18650 battery (3.7V)
- Reverse battery protection is done using a diode



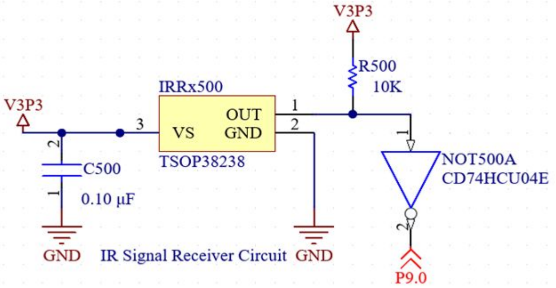


Hardware Summary - IR

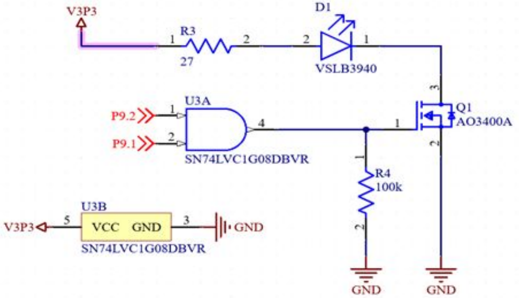
- IR Receiver Diode: TSOP38238 with a built-in carrier frequency of 38Khz
- AND pwm signal with TX signal into mosfet's gate input
- Attach pull up resistor to receiver's output since it is active low and then NOT this output



Receiver



Transmitter



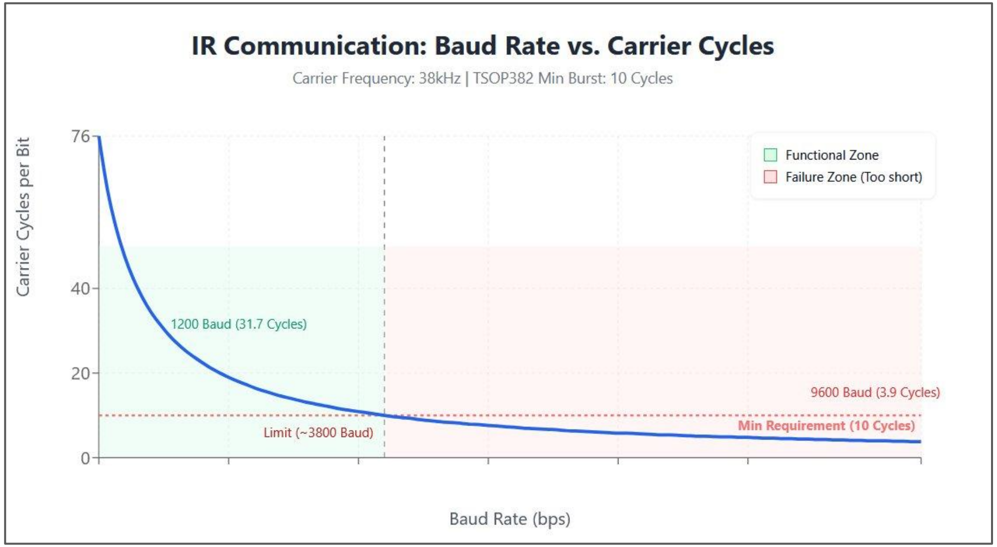
Hardware Summary - IR

	TSOP382..
Minimum burst length	10 cycles/burst
After each burst of length a minimum gap time is required of	10 to 70 cycles ≥ 12 cycles
For bursts greater than a minimum gap time in the data stream is needed of	70 cycles > 5 x burst length
Maximum number of continuous short bursts/second	1700

Baud rate = how many bits per second

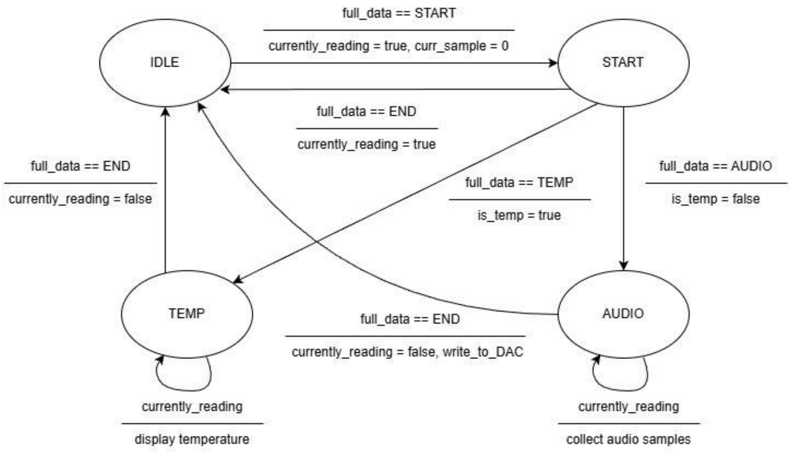
Carrier frequency = how many carrier cycles per second

Carrier Cycles Per Bit = f_carrier / baud_rate



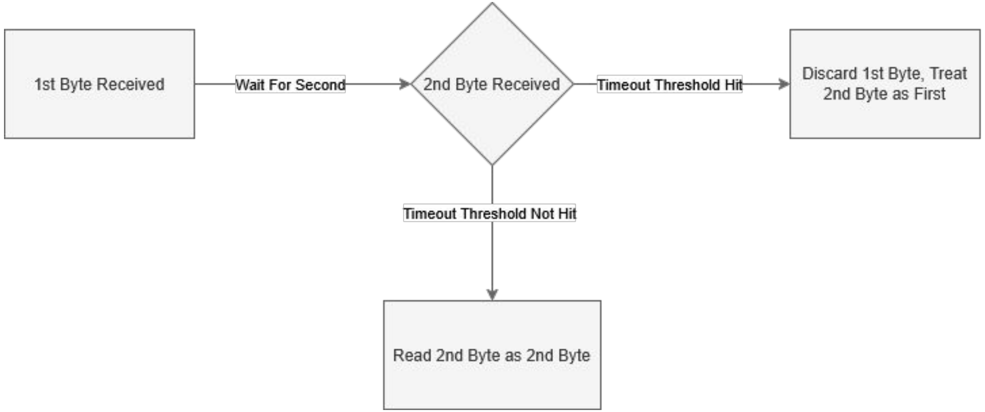
Software Summary - IR

- State Machine (Transmitting / Receiving Data)
    - To differentiate between signals/transmissions: Data packets: START, AUDIO, TEMP, END
    - Timer was used to determine if data needed to be sent
    - Sent data detected on TX pin P9_1 via UART
    - Continuously check if there is data in RX pin P9_0
    - Extract data out of UART object



Software Summary - IR

- Error Correction
  - To account for lost packets, software didn’t check for strict length, only end packet
  - End packet sent multiple times to increase chance reception
  - Also helped with half-byte alignment
  - Timeout function ultimately solved half-byte alignment issue
  - Final hardware is resistant to errors and can self-recover



Challenges / Results

- We were able to achieve all of our desired goals for this project
- We learned about these challenges, what could be done to address them:
  - How carrier frequencies work, what they do and how to use them
  - How to optimize audio sampling for both quality and data size
  - What makes a protocol resistant to errors and data loss
  - What the challenges of keeping 2 devices in sync are
  - How to design power supplies for sensitive electronics
  - What goes into building a form of wireless communication
  - How to work around physical limitations for good end user experiences