IRHUB

UCSB CE Capstone
Fall 2016 / Spring 2017

Jeremiah Prousalis, Nathaniel Bradley, Jesus Castro

DEVELOPMENT TEAM

Jeremiah Prousalis:

- Project Lead
- Firmware

Nathaniel Bradley:

- Hardware Lead
- Signal Processing

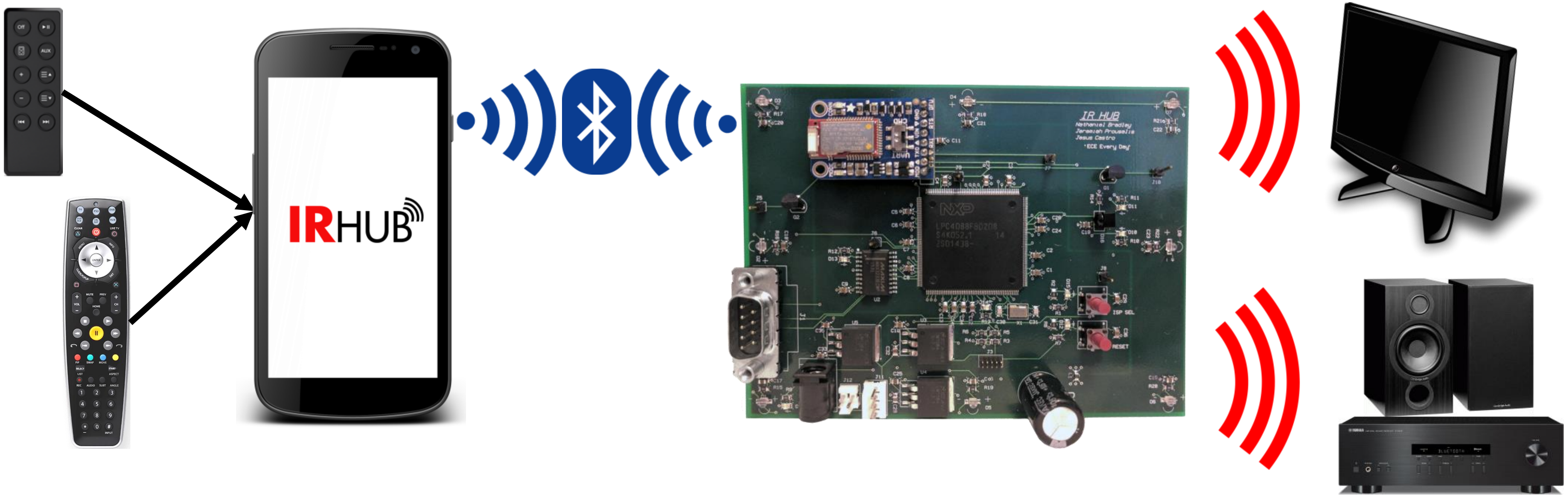
Jesus Castro:

- Software Lead
- Android Application

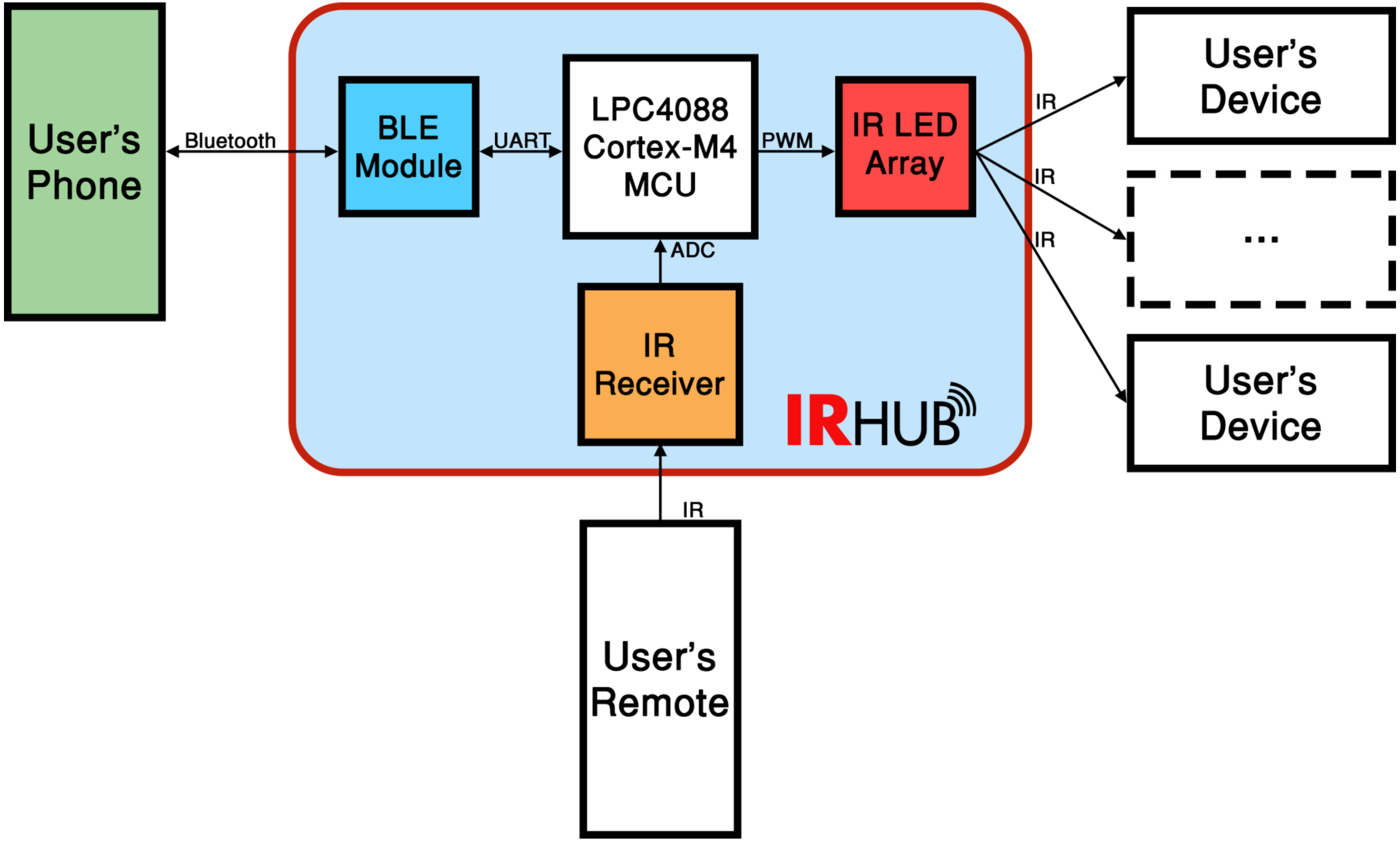
WHAT IT IS

IRHub is a device that turns your smartphone into a **Universal Remote** by combining three systems:

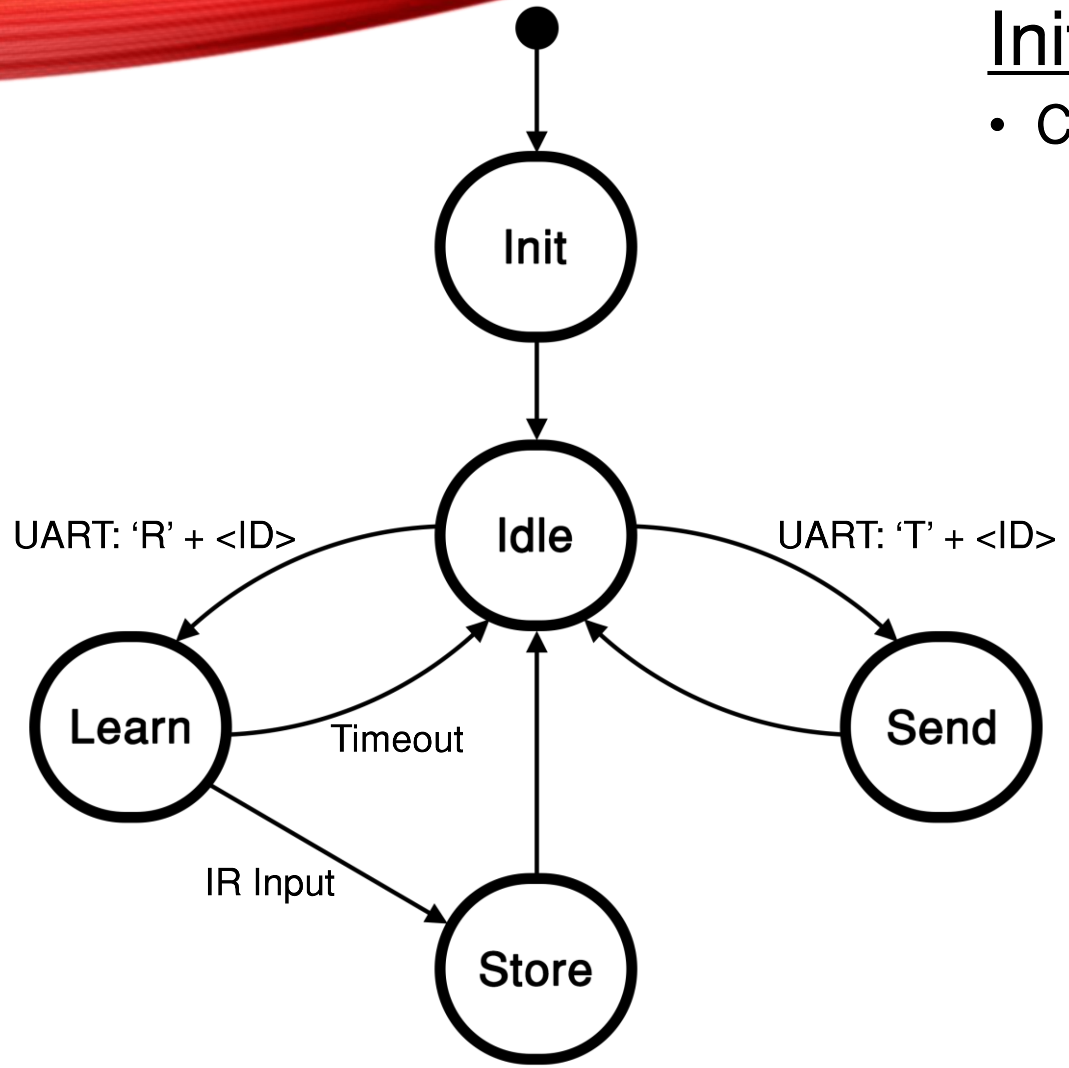
1. An **Infrared Receiver** to learn codes from any remote you have
2. An array of **IR LEDs** to transmit those codes with 360° room coverage
3. An **Android Application** to control the Hub over a Bluetooth connection



BLOCK DIAGRAM



STATE DIAGRAM



Initialization

- Copy previously stored remote codes from flash to RAM

Idle

- Wait on UART for command from phone

Learn

- Wait on ADC for input from remote

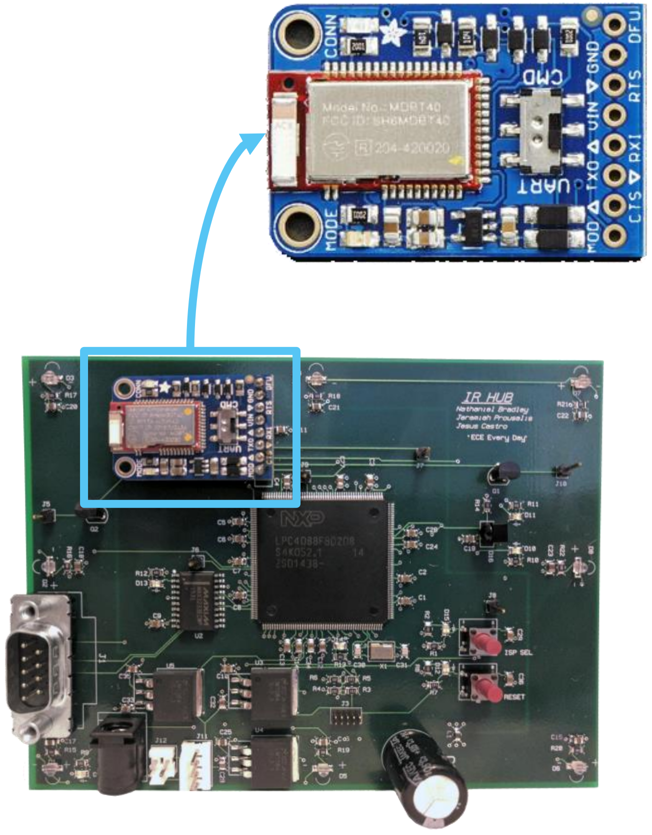
Store

- Decode signal and copy new code to flash with offset determined by <ID>

Send

- Access then transmit code in RAM at index of <ID>

BLUETOOTH CONNECTION



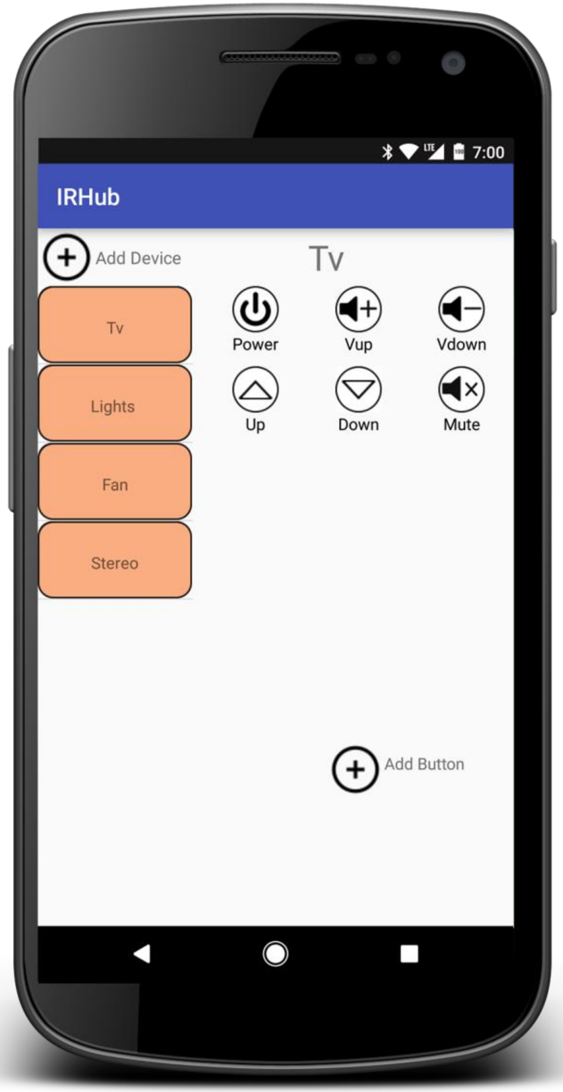
Adafruit BluefruitLE UART Friend

- Nordic UART connection profile offers transparent data pipe between Android’s Bluetooth connection and MCU’s UART
- Uses Bluetooth Low Energy (BLE) to minimize power consumption

Nordic UART Service

- **TX Characteristic**
  - Read Hub state feedback via this characteristic
- **RX Characteristic**
  - Write one of two commands to Hub via this characteristic:
    - “R” + <ID>: Read remote signal & store code at index of ID
    - “T” + <ID>: Transmit code stored at index of ID

ANDROID APPLICATION



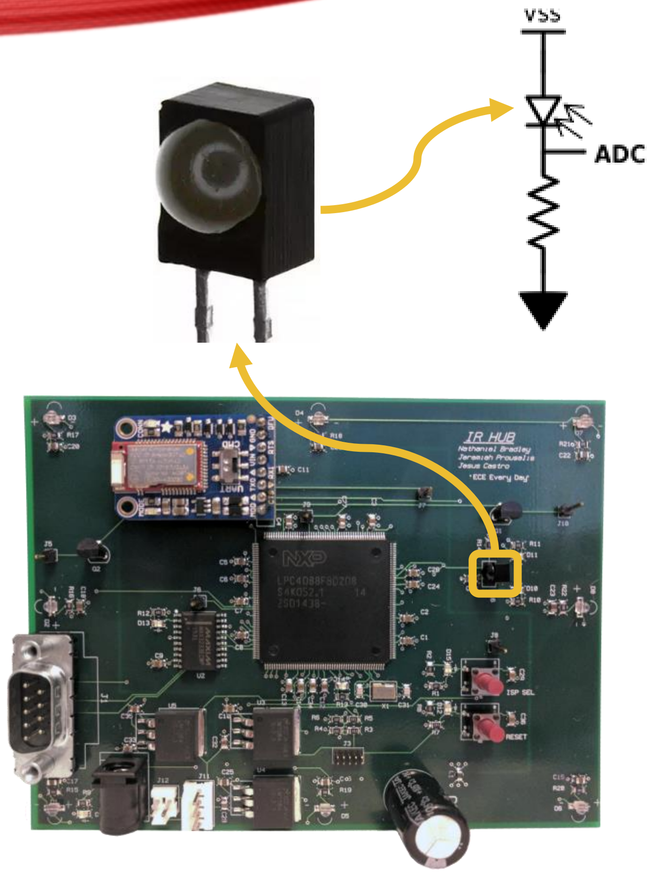
Organize Devices and Buttons

- Buttons on the App are grouped by device
- Users may add buttons they wish to control to the app
  - Common button names/icons available, but users may enter custom ones
- Signal codes for buttons are *not* stored on the App
- The App stores buttons associated with unique ID

Control IRHub

- When button is added, available ID and “Learn” command are sent to the Hub
- When button is pressed, associated ID and “Transmit” command are sent to the Hub

LEARN



940nm Wavelength Photodiode

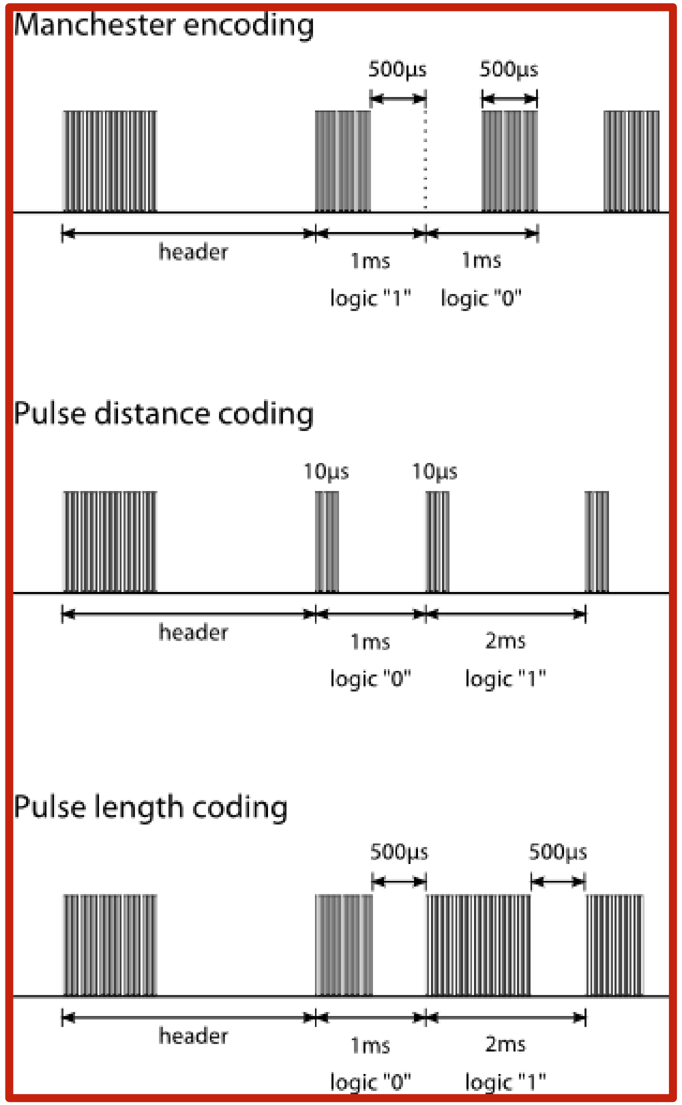
- Sensitive to same IR wavelength as those found in IR remotes
- During “Learn” state, MCU waits for input on 12-bit ADC
    - Signal edge triggers ADC sample at rate of 200 kHz
    - Sample is decoded and stored in on-board flash memory
    - Up to 128 button signals can be stored simultaneously

IR PROTOCOLS

Common Consumer Encoding Schemes

- Remote signals layered on carrier frequency
  - Carriers may be anywhere from 36-100kHz
  - Photodiode used over dedicated receiver module to support wider range of carriers

- Digital encoding schemes vary between manufacturers
  - Custom digital decoding scheme needed to keep support generic



SIGNAL DECODING

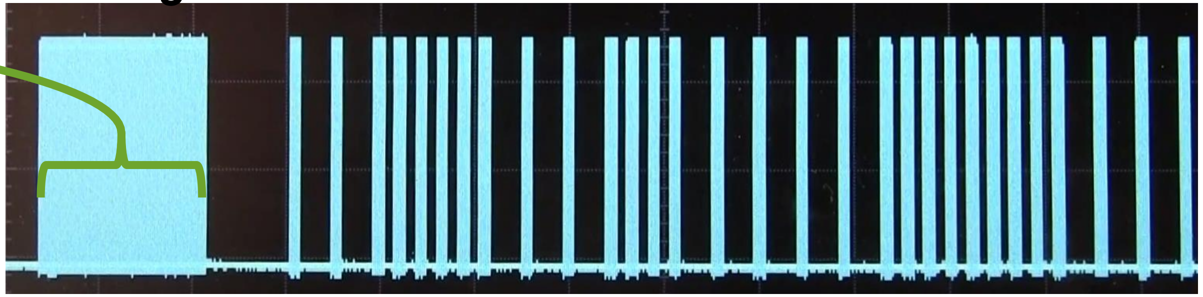
1. **Determine carrier frequency**
   - Use FFT to pull carrier out of raw input

2. **Determine duration** of every unique on or off period that appears in signal

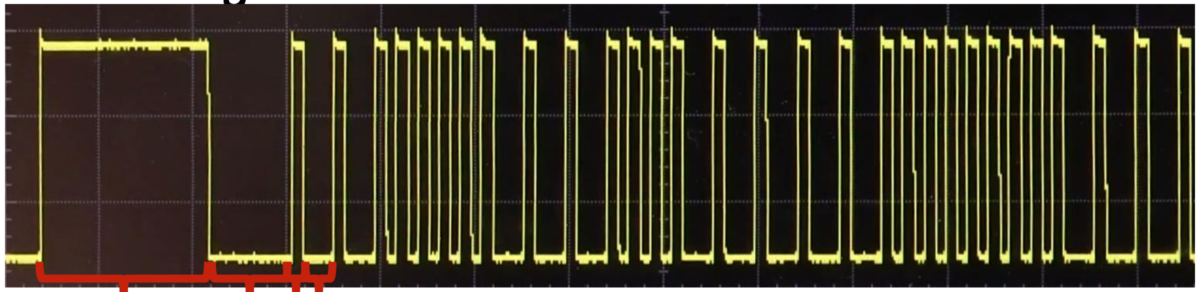
3. **Map sequence** of highs and lows to duration signal is high or low

4. **Store**: *Carrier Frequency*, *Duration Array*, and *Sequence Array*
   - Code is stored in flash memory at offset determined by ID provided by phone

Raw Signal:



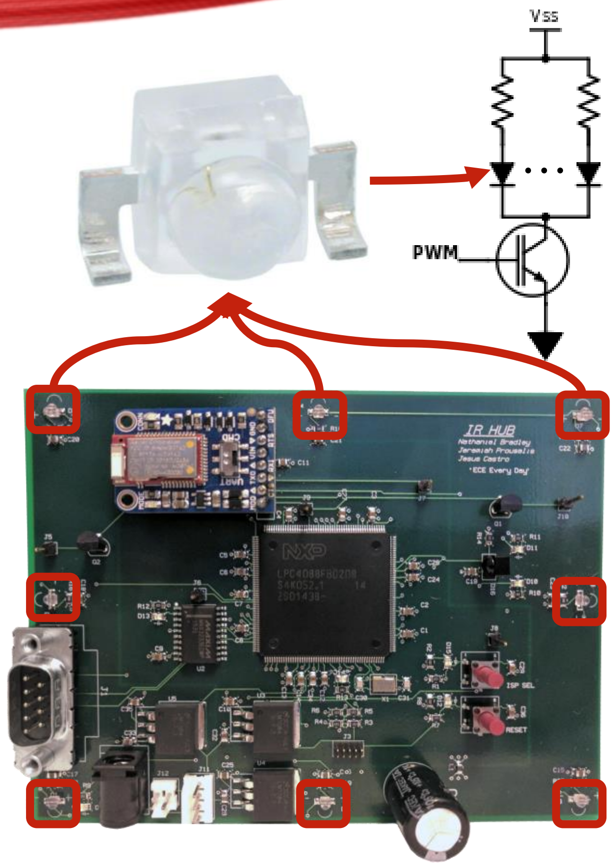
Filtered Signal:



Unique Pulse Duration (µs)	9180	4.320	560	1690
Index	[0]	[1]	[2]	[3]

On/Off Sequence	0	1	2	3	2	…

TRANSMIT



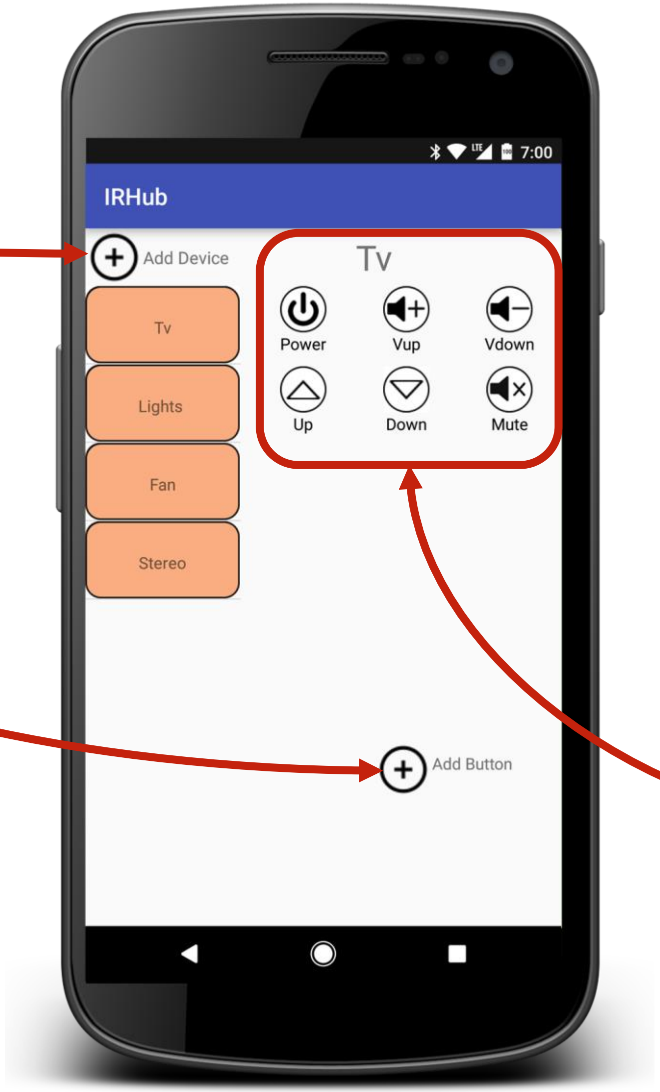
940nm Wavelength, 1.35V IR LEDs

- Same IR wavelength LEDs as those found in IR remotes
- 8 LEDs around perimeter broadcast code 360°
  - Positioning Hub in center of room allows signals to reach and control any devices within line of sight of board
- During "Transmit" state, PWM from MCU drives emitter array
  - Code from index ID accessed
  - PWM frequency set to *carrier* stored with code
  - *Sequence array* is iterated through
  - PWM is alternated on/off for time at index of *Duration array* pointed to by a given sequence slot

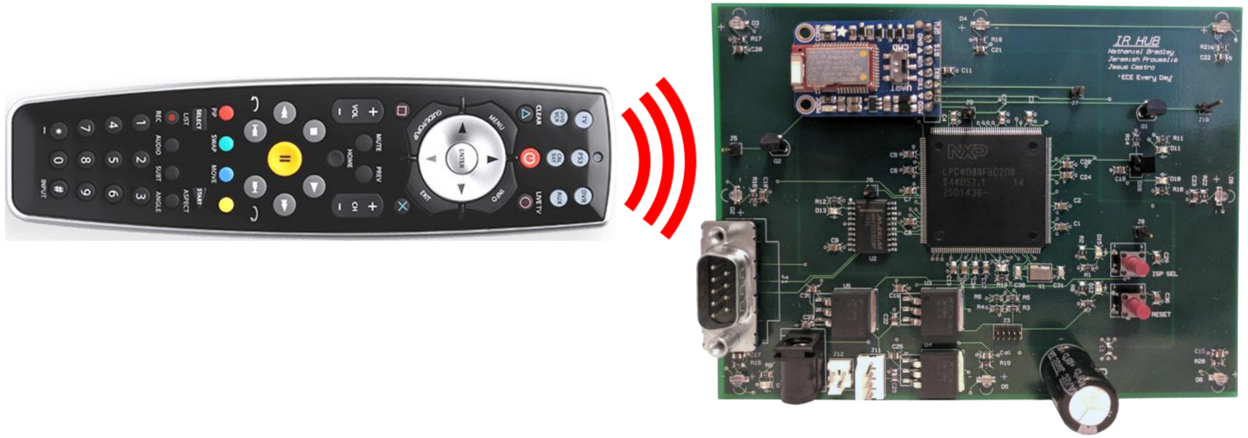
USING IRHUB

1) Add device you want to control and give it a name

2) Add and name desired button for that device

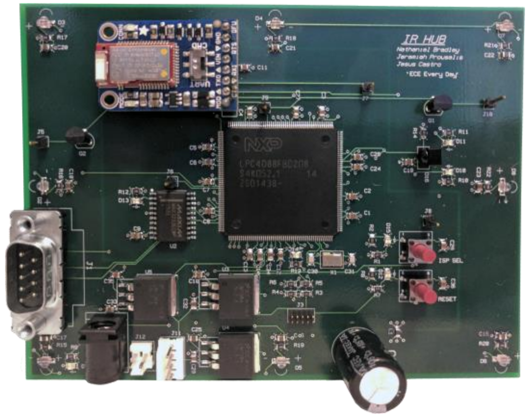


3) Point your remote at the Hub and press that button



4) Pressing that button on the app will now control your device

COST



Total Cost:

- All board components ($63.75 per board)

Primary Contributors:

- LPC4088 Cortex-M4 MCU ($12.92)
- Adafruit BLE Module ($17.50)

Potential Cost Reduction

- Remove components only needed for development
  - $9.40 in parts are not needed for functionality
- Single MCU to replace LPC4088 and BLE Module
  - NRF51822 Cortex-M0 MCU (Under $4)

KEYS TO CAPSTONE SUCCESS

Give yourself options

- It’s better to have stuff you don’t need than need stuff you don’t have
  - Original plan did not demand the DSP capabilities of the Cortex-M4
  - Redundant paths on our board made several methods for reading signals available
  - Still many “I wish we did…” moments

Stay on track

- Finish Milestones early to have extra time to triple check your work
- Mistakes are made when you fall behind
  - A hurriedly placed sensor gave us problems before we realized it was on backwards

ACKNOWLEDGEMENTS

Instructors:

- John Johnson and Yogananda Isukapalli

Teaching Assistants:

- Celeste Bean, William Miller, Caio Motta

Others:

UCSB

Mentor Graphics®

NXP

SUNSTONE® CIRCUITS

LARITECH, INC. ODM SOLUTIONS

Embedded Artists

RAPID PROTOTYPES TECHNOLOGY & MANUFACTURING, LLC