Lecture 22

HTC Vive Tracking System
UDP review

Stuff

- Team assignments should be finished by the end of this week. Individuals not on a team will be assigned.
  - 4-member teams can join team 38,39,40
- T-shirt order reminder (about 30 people haven't responded)
  - See piazza post for link (search t-shirt and pinned)
- Purchasing policy: returns – exchanges – not allowed. Part of your project budget can go towards mistaken purchases.

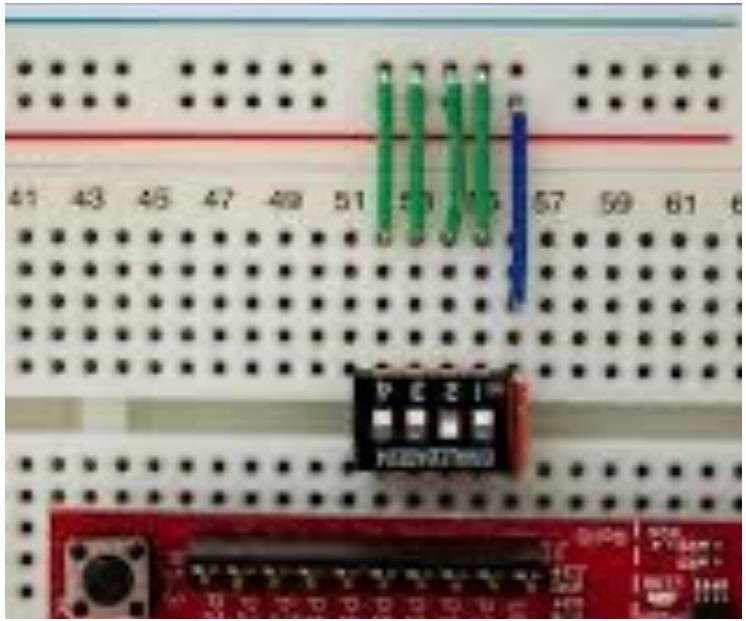
Game Issues

- New Game Rule: Scoring objects must be touching the ground in the doubling square to be doubled.
  - Explicitly allowed: picking up and holding/storing cans – but the top of the can must be exposed to Vive.
- New Game Rule: All robots must have ability to quickly set robot ID to 1,2,3 or 4 depending on red/blue team assignment (to be sent as part of UDP packet).

- Final Project Scheduling Vote – winner is…
- Lab4 weighting in class grade reduced from 20% to 15%
- Final Project weighting increased from 30% to 35%

Setting Robot ID

- Before each game, robots will need to set their ID. Red side robots 1 and 2, Blue side robots 3 and 4.
- Easy way use DIP switches (available in GM lab)
- Can use `pinMode(#,INPUT_PULLUP);`



Don't rely on uploading code just before a match.

Agenda

01. What is the HTC Vive Tracker
02. Signal conditioning for the Vive
03. Using UDP for Final Project

How this lecture relates to the game and your graded evaluation:

Graded Performance Requirement:

- Your robot must determine its XY location using Vive.
  - You must make a vive reading circuit on your robot.
- Your robot must transmit this XY location using UDP broadcast packets once per second.
  - You must incorporate code to periodically transmit via UDP

Note:

- Your robot does not necessarily need to *receive* UDP packets, but it's likely to be useful for the game.

01

HTC Vive Trackers

VR Object Tracking in 3 Space

https://www.youtube.com/watch?v=Dw8La6fylLQ

Tracker devices on hands and feet. Track 3D position in real time.

Lighthouse Sweeping the Room

https://www.youtube.com/watch?v=1Ldilsh4_Gk

- When I point to you, remember the number you hear from the video. We'll do it twice, once for X, once for Y.

Q1 What are your coordinate in the room?

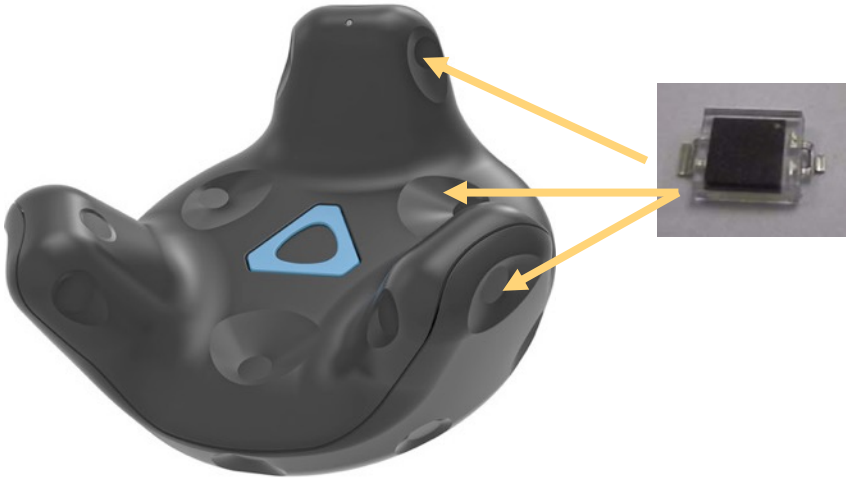
- Where is the information about your location stored – in the sweeper? Or the sweepee?
  - Distributed self-localization

Q2 What components hardware do we need to implement this type of coordinate localization?

- Base station - sweeper
- Timer
- Something to sense when sweep is seen
- Something to sense when timer should start (could be same sensor)

HTC Vive trackers:

- Base stations does the light sweeping
- Photosensor senses initial flash and sweeping light
- Timer measures time between flash and sweep
- Triangulate from multiple base stations

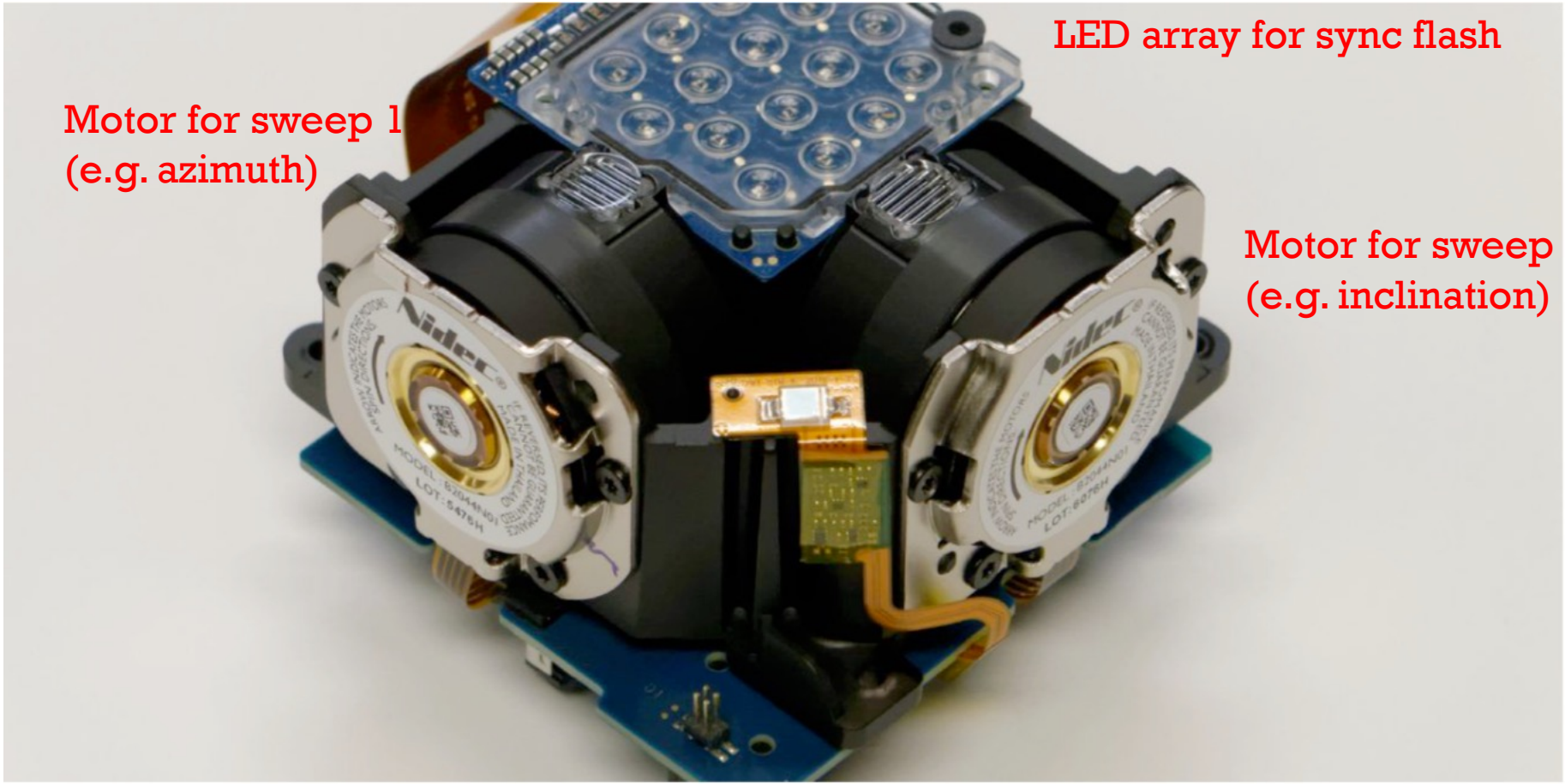


HTC Vive Light House

Sync flash

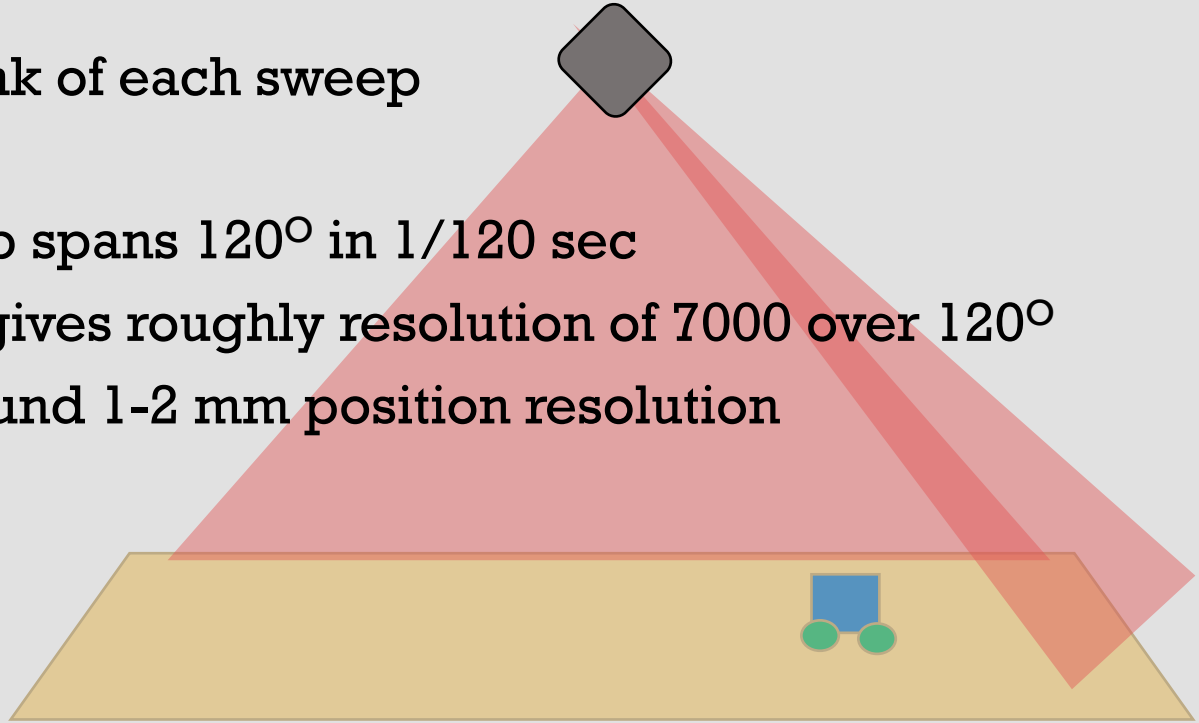
https://www.youtube.com/watch?v=tSXbJQO2lJE

Disected vive base station



Use in 510 final competition.

- We can think of each sweep as X and Y
- Each sweep spans 120° in 1/120 sec
- Our timer gives roughly resolution of 7000 over 120°
- Expect around 1-2 mm position resolution



Mounted in GM Lab

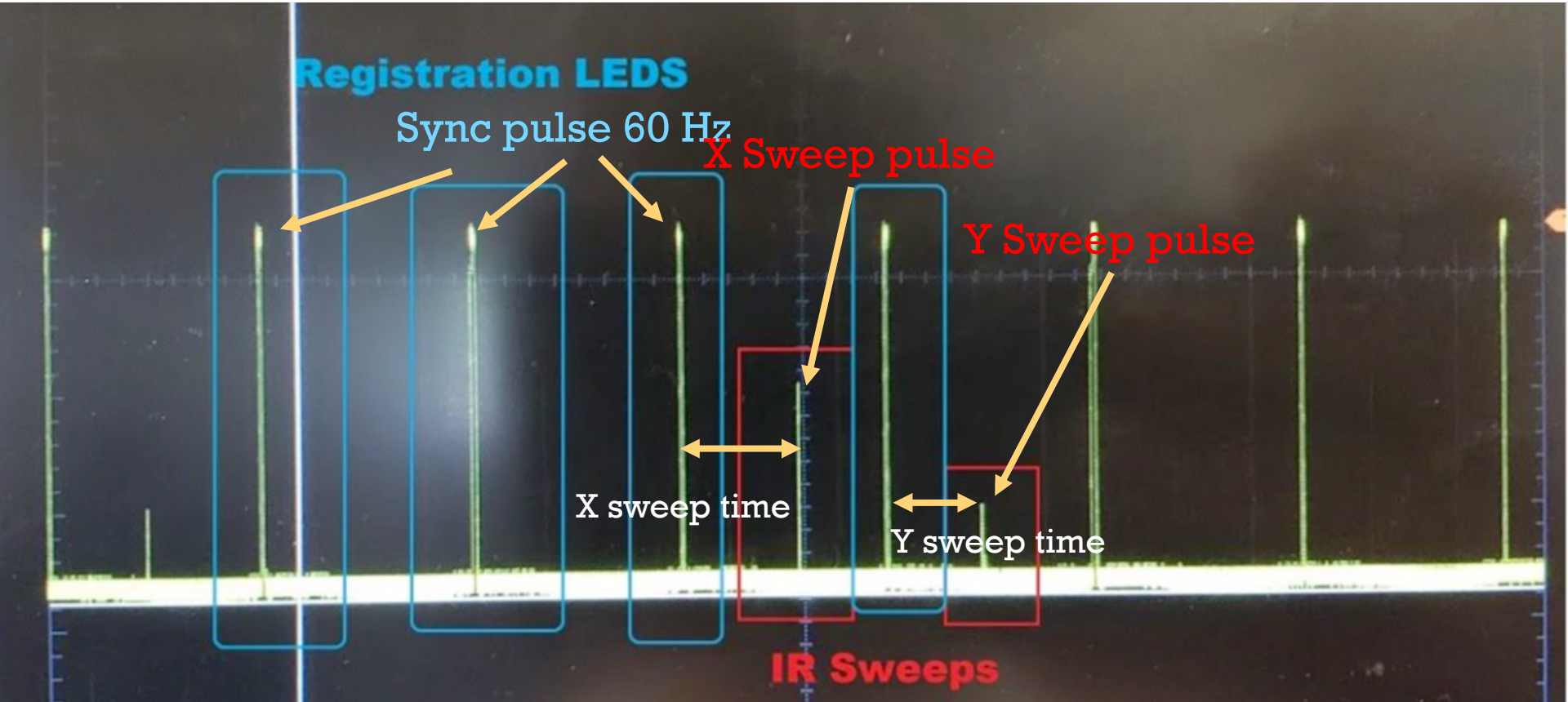
- Plug it in to start.
- Unplug at end of day.

Vive mounted here

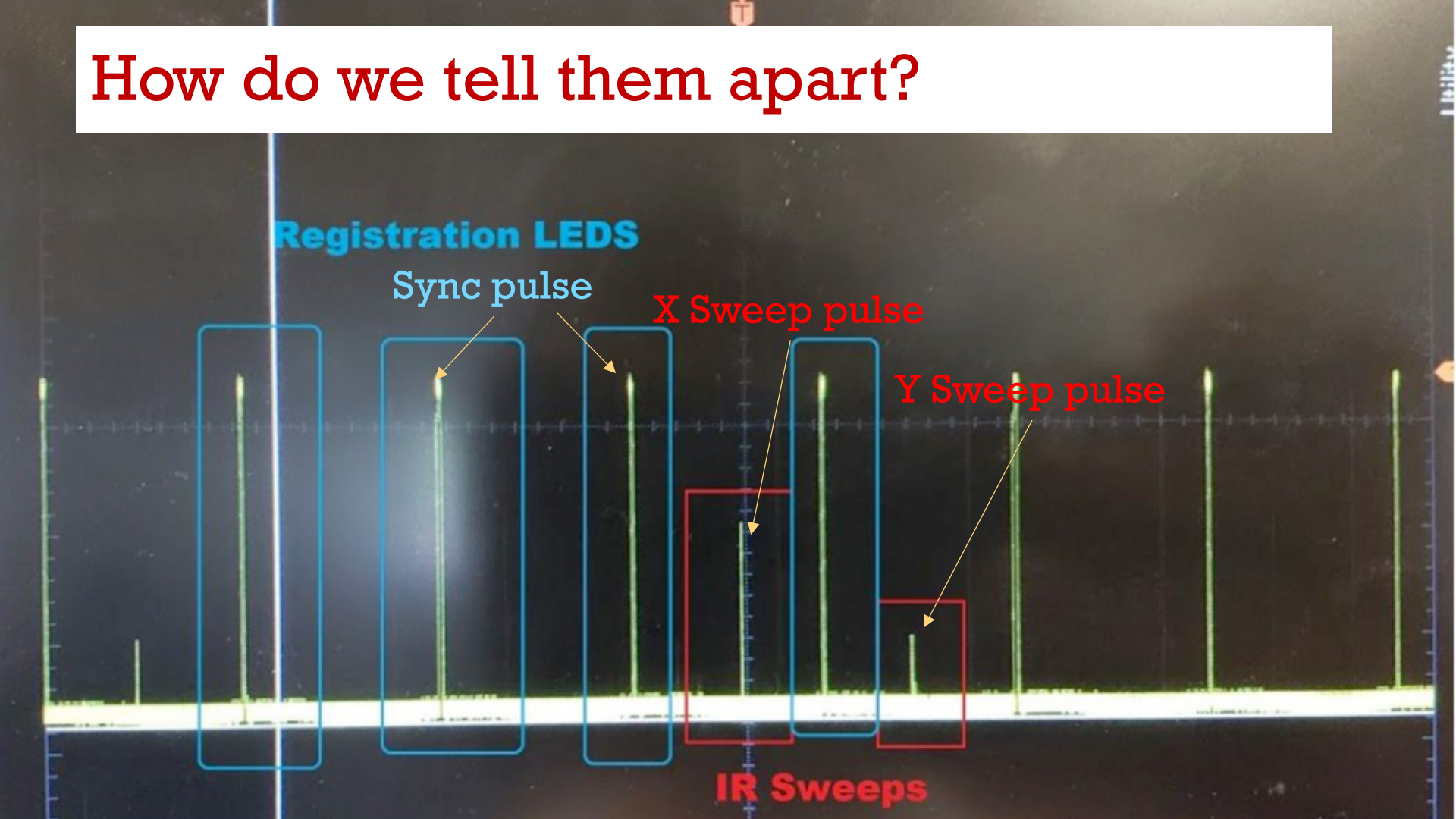
Vive plugged in here

- When running, look for green light "A" and pattern of 3x3 red dots

Light signal. Registration and sweeps occur 60 times per sec

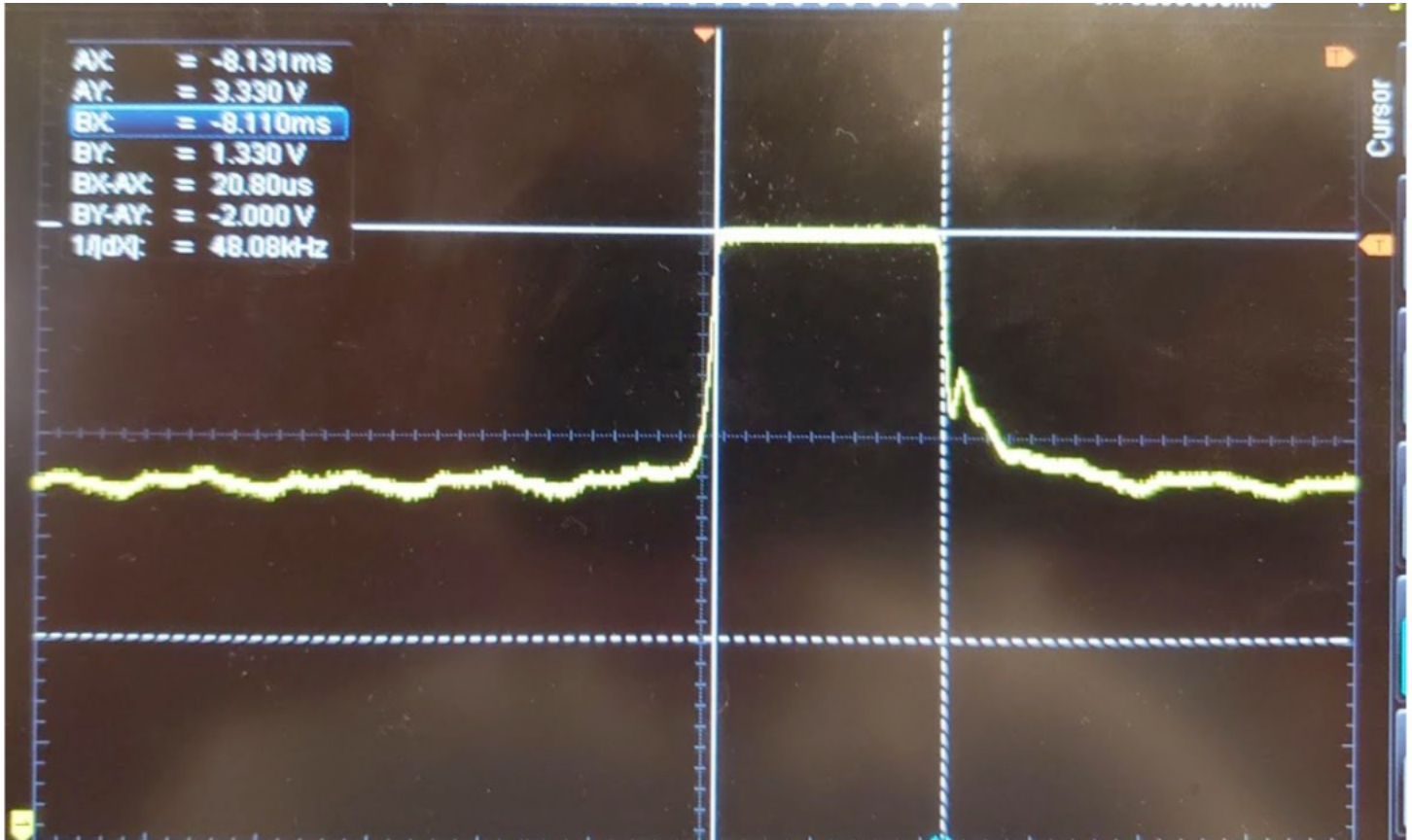


How do we tell them apart?



Sync and sweep pulse width

- Amplitudes will vary
- Sync flash has width about 104 - 135uS
- Sweep has width about 30uS

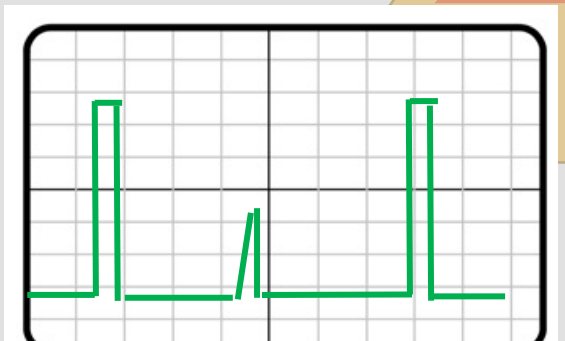


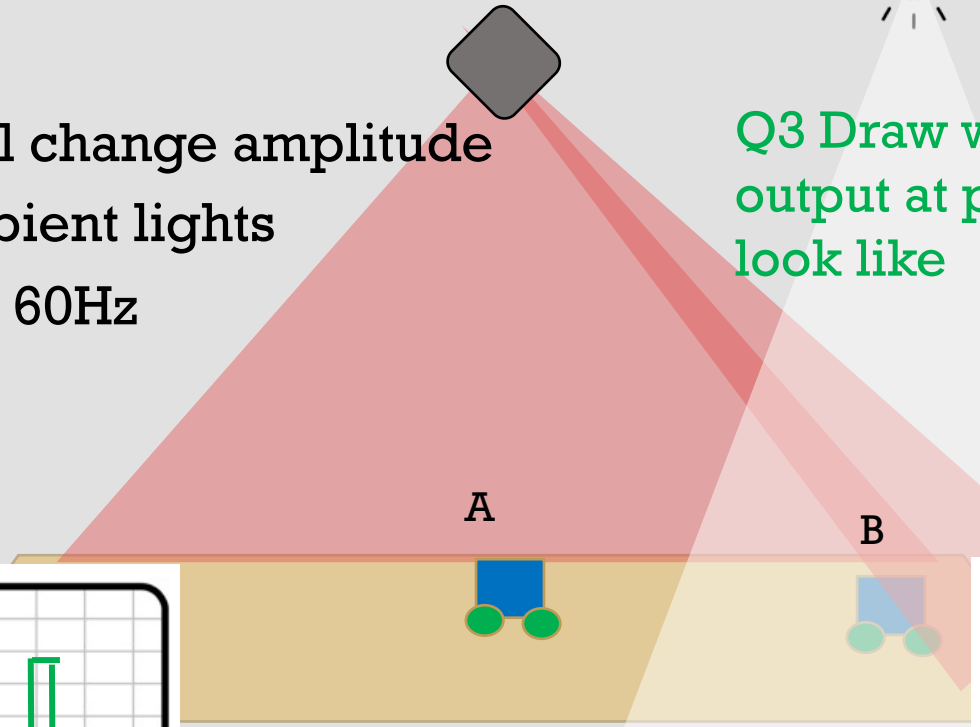
02

Signal conditioning for Vive

Signal variance

- Location will change amplitude
- Varying ambient lights
- AC lights @ 60Hz

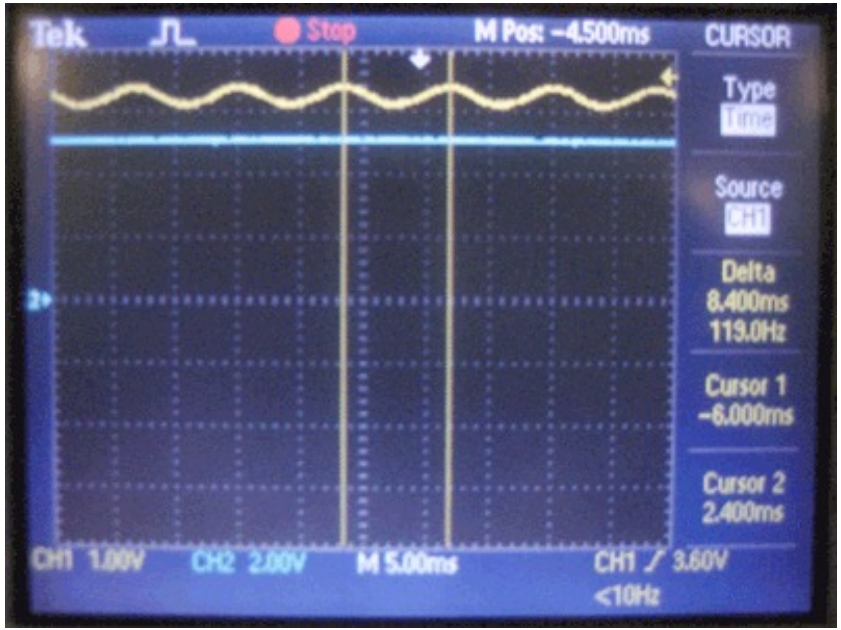




Q3 Draw what the sensor output at point B might look like

How can we make our circuit more robust to ambient light?

- Lights are run off of AC power at 60Hz in the U.S.

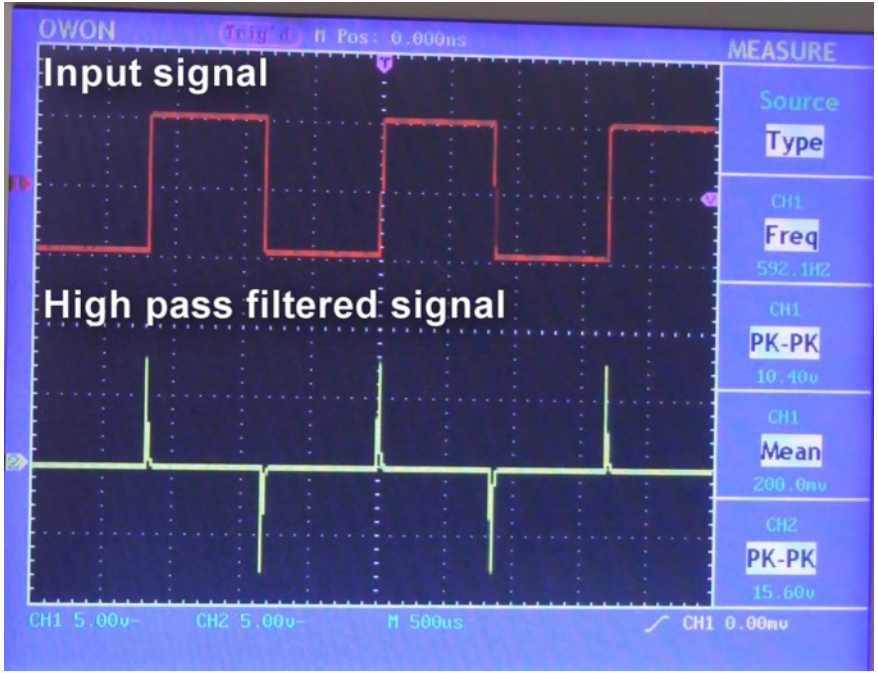


How can we make our circuit more robust to ambient light?

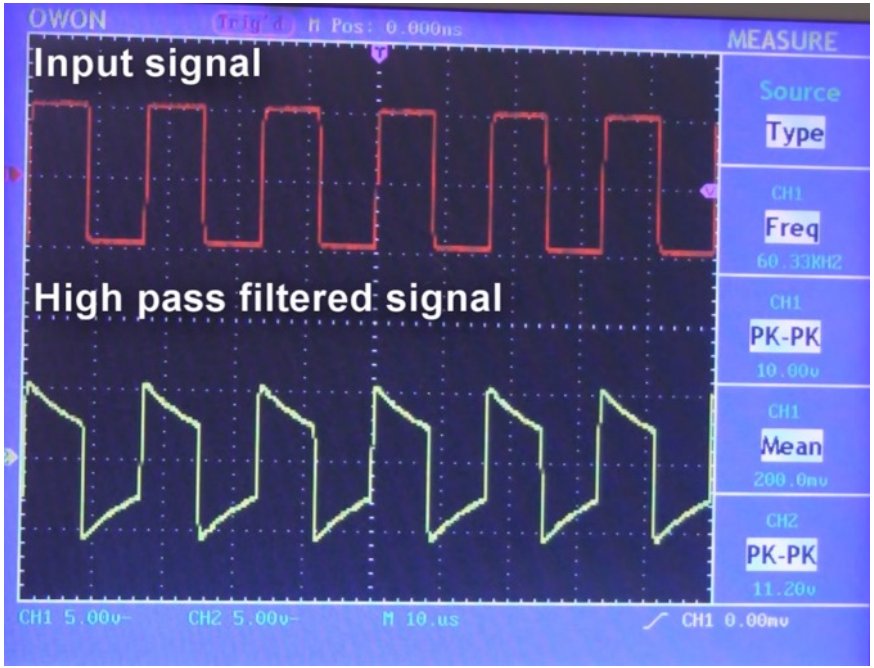
- Q4 What types of filter could we use?

- Q5 What are the important frequencies to consider?

High pass filter on a square wave



Square wave freq << corner freq of filter



Square wave freq = 2x corner freq of filter

https://www.youtube.com/watch?v=4CcIFycCnxU

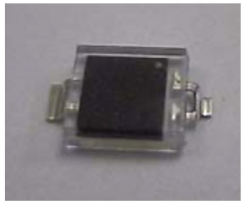
Detecting pulse

Q6 Will GMlab IR phototransistor LTR-4206E work for our system?

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT	TEST CONDITION	BIN NO.
Collector-Emitter Breakdown Voltage	$V_{(BR)CEO}$	30			V	I_C = 1mA Ee = 0mW/cm²	
Emitter-Collector Breakdown Voltage	$V_{(BR)ECO}$	5			V	I_E = 100 μA Ee = 0mW/cm²	
Collector Emitter Saturation Voltage	$V_{CE(SAT)}$			0.4	V	I_C = 0.5mA Ee = 1mW/cm²	
Rise Time	Tr		10		μs	V_{CC} = 5V I_C = 1mA R_L = 1KΩ	
Fall Time	Tf		10		μs		
Collector Dark Current	I_{CEO}			100	nA	V_{CE} = 10V Ee = 0mW/cm²	
On State Collector Current	$I_{C(ON)}$	0.4		1.2	mA	V_{CE} = 5V Ee = 1mW/cm² λ =940nm ~Full daylight	BIN B
		0.8		2.4			BIN C
		1.6		4.8			BIN D
		3.2		9.6			BIN E
		6.4					BIN F

PD70-01C IR Photodiode (speed?)

PD70-01C/TR7



- Used in Vive products so we know it works.
- Everlight Data sheet does not show speed!
- But, websites that sell them do.
- PD70-01C has response time 50nS

Features

- High sensitivity
- Low capacitance
- Short switching time
- Wide temperature range
- Small package
- Pb free
- The product itself will remain within RoHS compliant version.
- Compliance with EU REACH.
- Compliance Halogen Free .(Br <900 ppm ,Cl <900 ppm , Br+Cl < 1500 ppm).

Descriptions

- The PD70-01C/TR7 is high sensitivity, fast switching times, low capacitance, compact size, and lack of measurable degradation make it suitable for diverse applications, such as TV and appliance remote control, IR sound transmission, video recorders, and measurement and control.

Signal Conditioning OpAmps

MCP6041/2/3/4

Q7) For a 30nS pulse, which OpAmp might work better?

AC ELECTRICAL CHARACTERISTICS

Electrical Characteristics: Unless otherwise indicated, V_{DD} = +1.4V to +5.5V, V_{SS} = GND, T_A = 25°C, V_{CM} = $V_{DD}/2$, $V_{OUT} \approx V_{DD}/2$, $V_L = V_{DD}/2$, R_L = 1 MΩ to V_L, and C_L = 60 pF (refer to Figure 1-2 and Figure 1-3).

Parameters	Sym	Min	Typ	Max	Units	Conditions
AC Response						
Gain Bandwidth Product	GBWP	—	14	—	kHz	
Slew Rate	SR	—	3.0	—	V/ms	
Phase Margin	PM	—	65	—	°	G = +1 V/V
Noise						
Input Voltage Noise	E_{ni}	—	5.0	—	μV_{P-P}	f = 0.1 Hz to 10 Hz
Input Voltage Noise Density	e_{ni}	—	170	—	nV/√Hz	f = 1 kHz
Input Current Noise Density	i_{ni}	—	0.6	—	fA/√Hz	f = 1 kHz

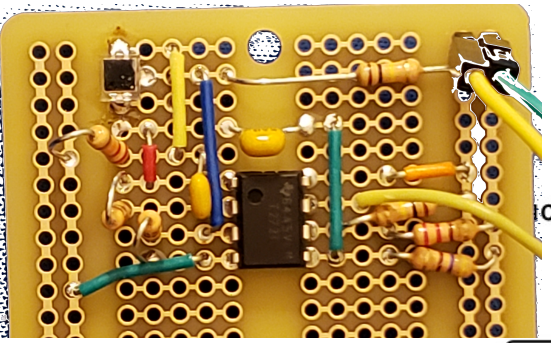
TLV272

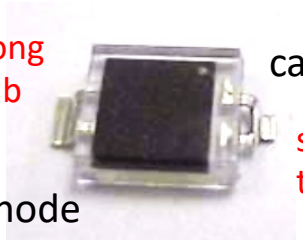
7.10 Electrical Characteristics: Dynamic Performance

over operating free-air temperature range (unless otherwise noted).

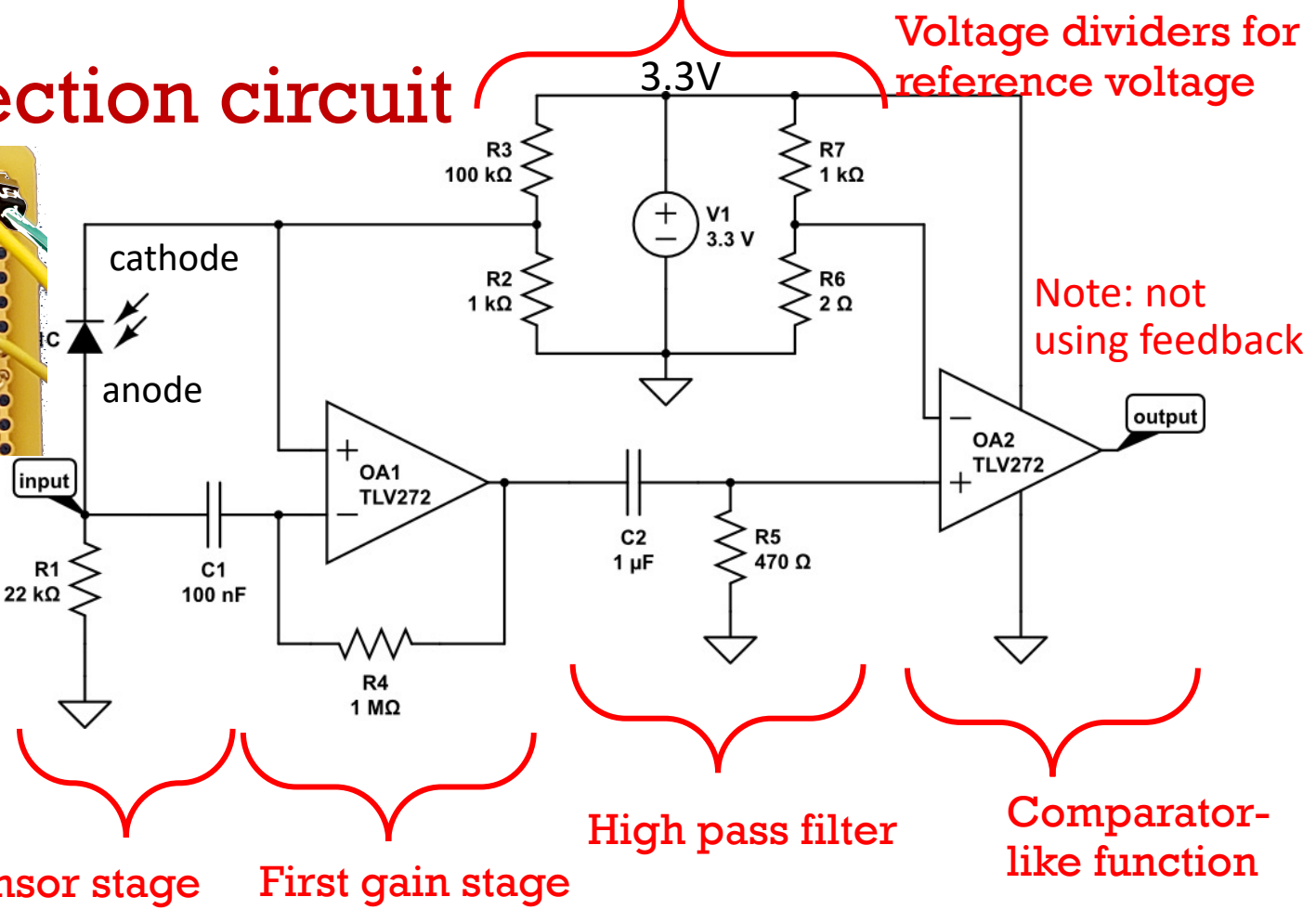
PARAMETER		TEST CONDITIONS		T_A (1)	MIN	TYP	MAX	UNIT
UGBW	Unity-gain bandwidth	R_L = 2 kΩ, C_L = 10 pF	V_{DD} = 2.7 V	25°C		2.4		MHz
			V_{DD} = 5 V to 10 V	25°C		3		
SR	Slew rate at unity gain	$V_{O(PP)} = V_{DD}/2$, C_L = 50 pF, R_L = 10 kΩ	V_{DD} = 2.7 V	25°C	1.35	2.1		V/µs
				Full range	1			
			V_{DD} = 5 V	25°C	1.45	2.4		V/µs
				Full range	1.2			
			V_{DD} = ±5 V	25°C	1.8	2.6		V/µs
				Full range	1.3			

Vive detection circuit



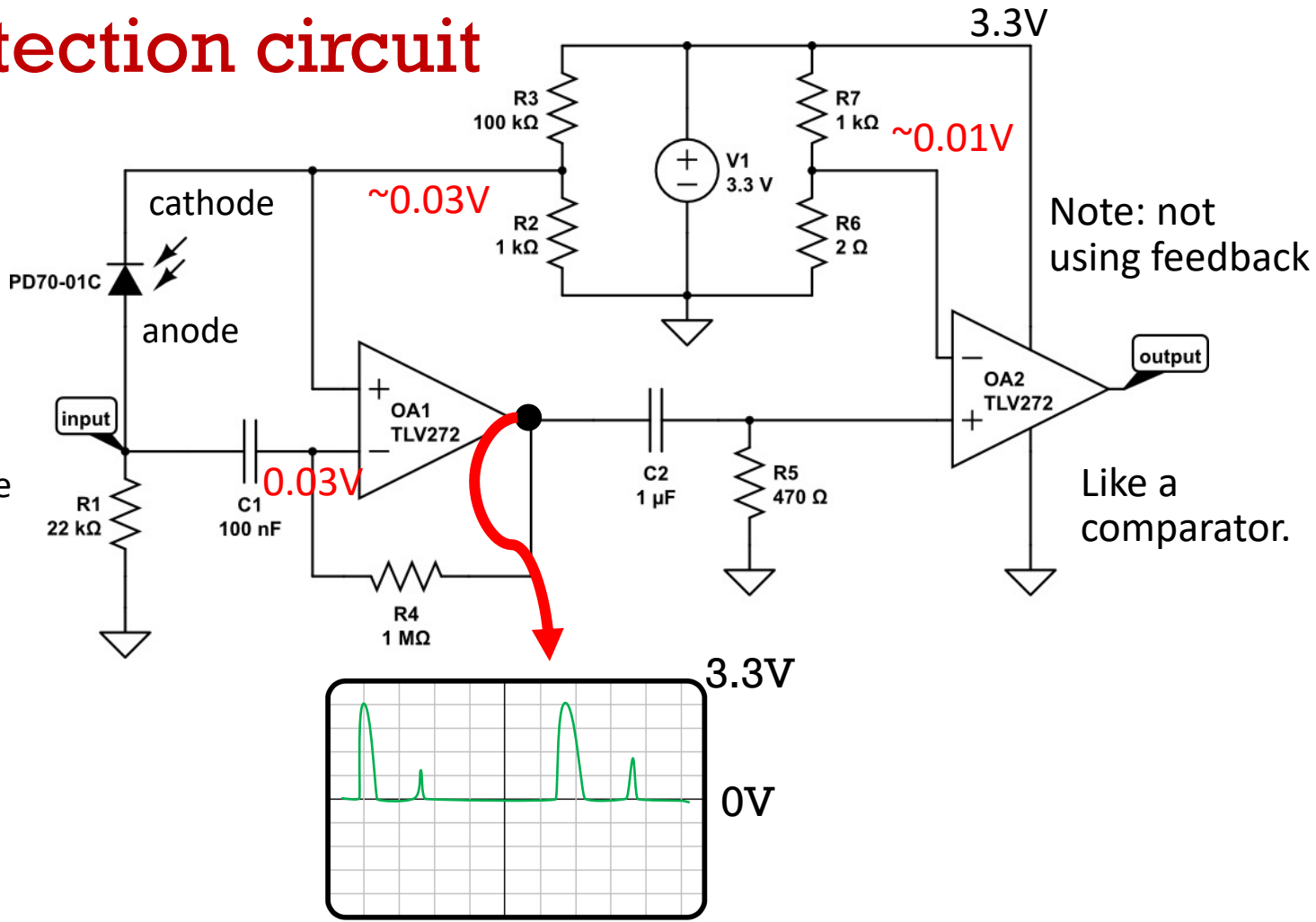


Small surface mount diode



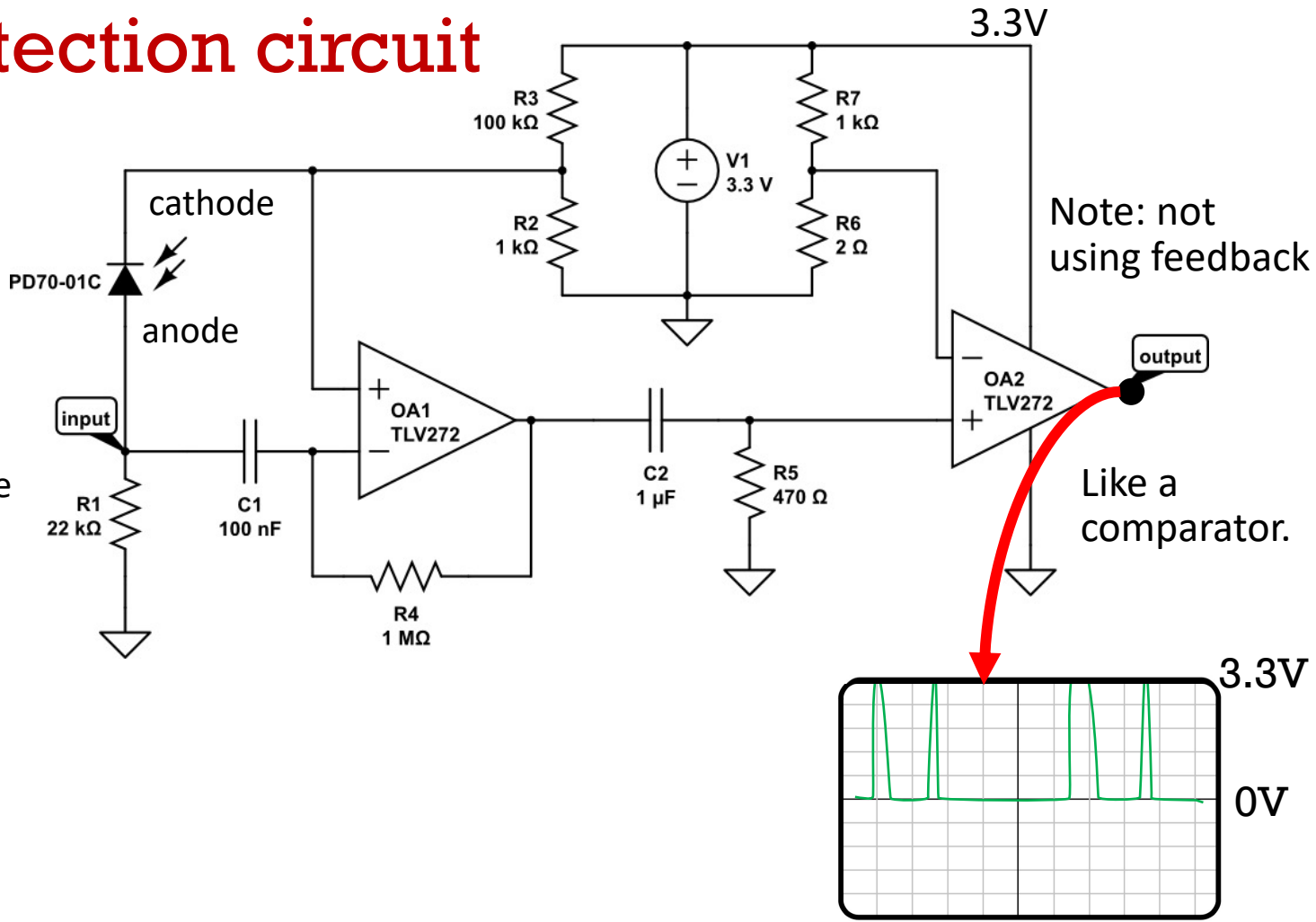
Vive detection circuit

Small surface mount diode



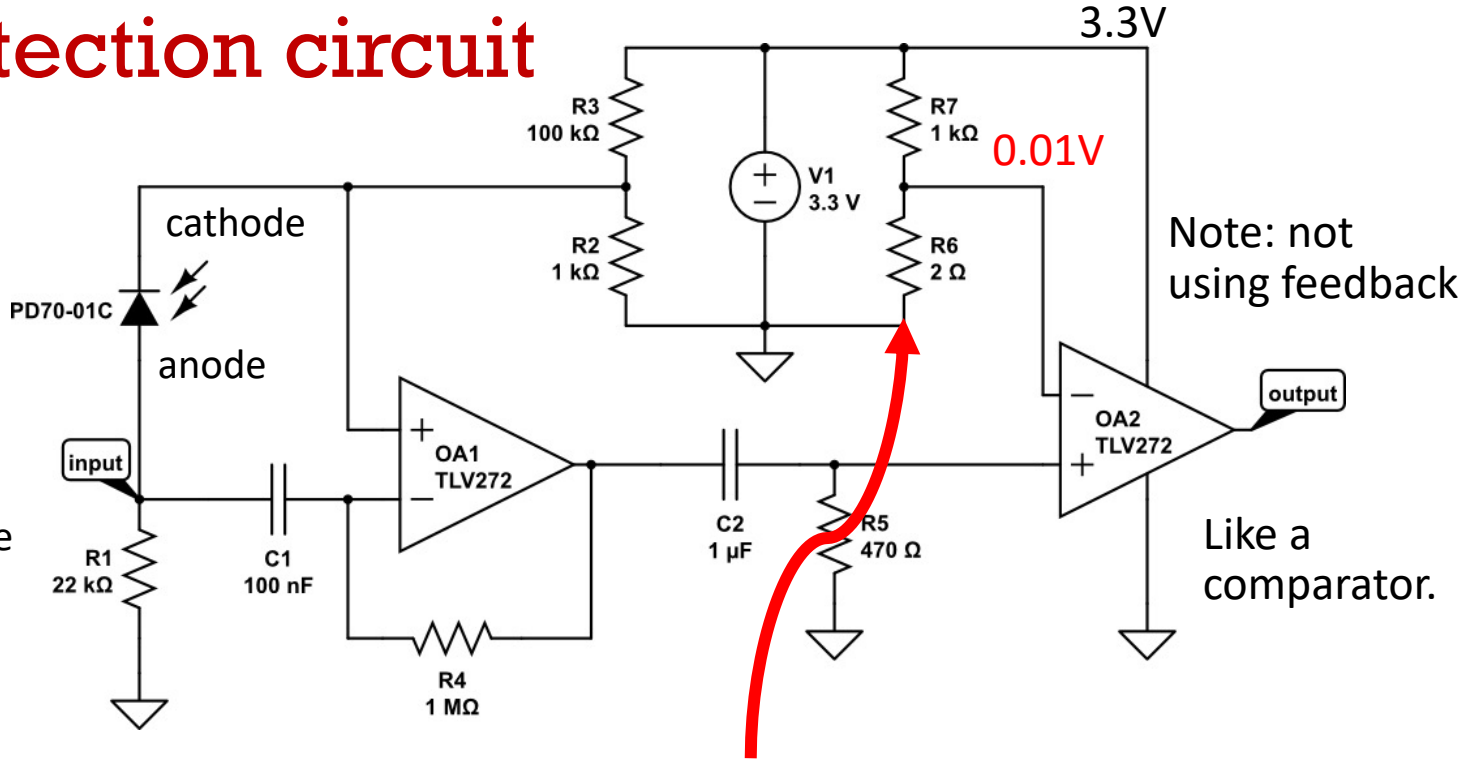
Vive detection circuit

Small surface mount diode



Vive detection circuit

Small surface mount diode



Note: not using feedback

Like a comparator.

Change resistors to change sensitivity. Higher ref voltage will be less sensitive and more robust to noise

Vive510 commands

Setup Vive

Global

```
#include <vive510.h>
Vive510  vivename;
```

Goes in setup

```
vivename.begin(int GPIO_Pin#);
```

xCoord returns x, yCoord returns y.

Status locked on

```
if (vivename.status() == VIVE_LOCKEDON)
X = vivename.xCoord();
Y = Vivename.yCoord();
```

Sync to lock

Status not locked on

```
if (vivename.status() != VIVE_LOCKEDON)
    vivename.sync(int 15);
```

Vive software example [using vive510.cpp]

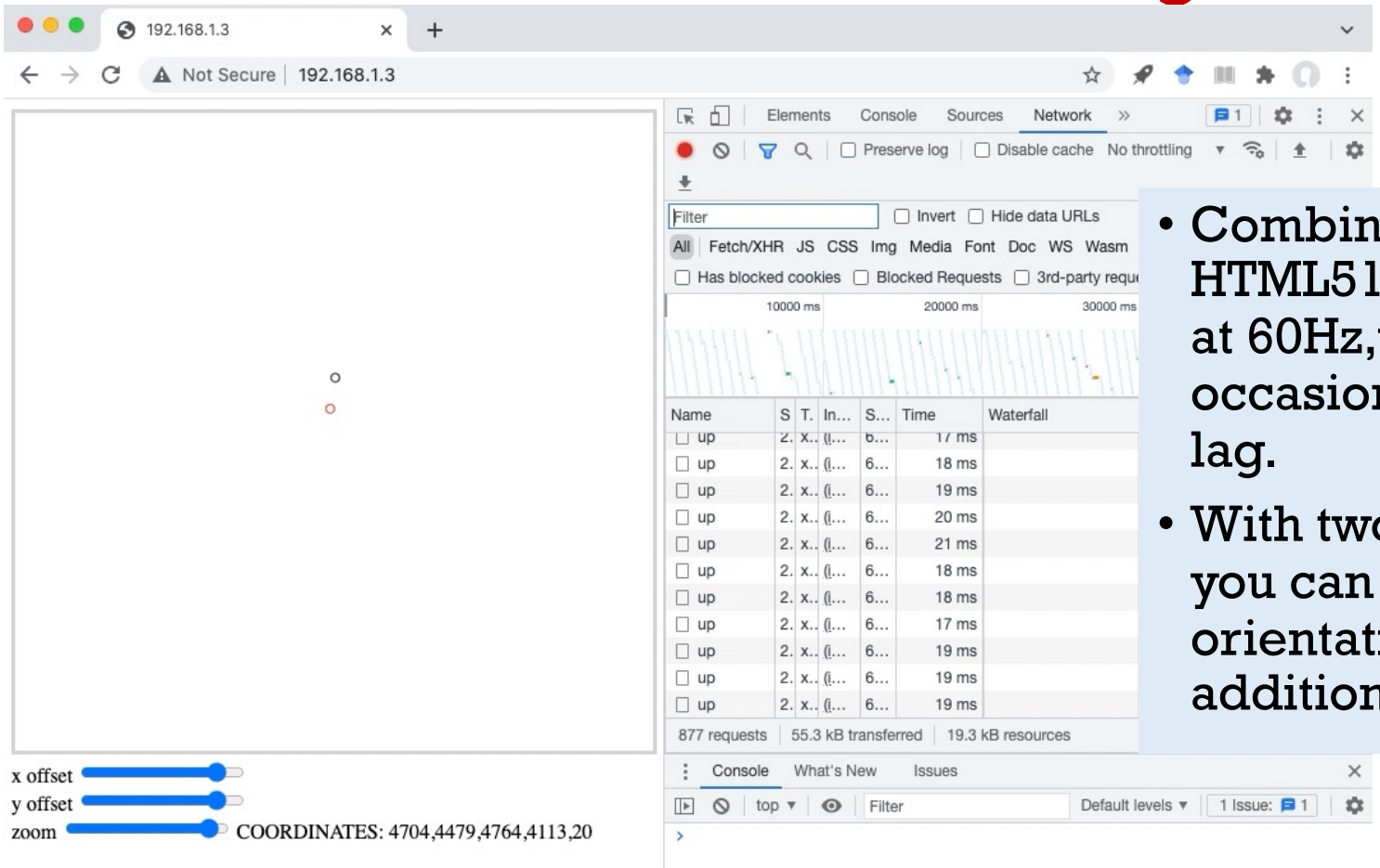
```
#include "vive510.h"
#define SIGNALPIN1 1   // GPIO pin receiving signal from Vive circuit
Vive510 vive1(SIGNALPIN1);

void setup(){
  Serial.begin(115200);
  vive1.begin();
  Serial.println("Vive trackers started");
}

void loop() {
  if (vive1.status() == VIVE_LOCKEDON) {
    Serial.printf("X %d, Y %d\n",vive1.xCoord(),  vive1.yCoord());
  }
  else
     vive1.sync(15);   // try to resync 15 times (nonblocking);
  delay(100);
}
```

On canvas resources vive510/

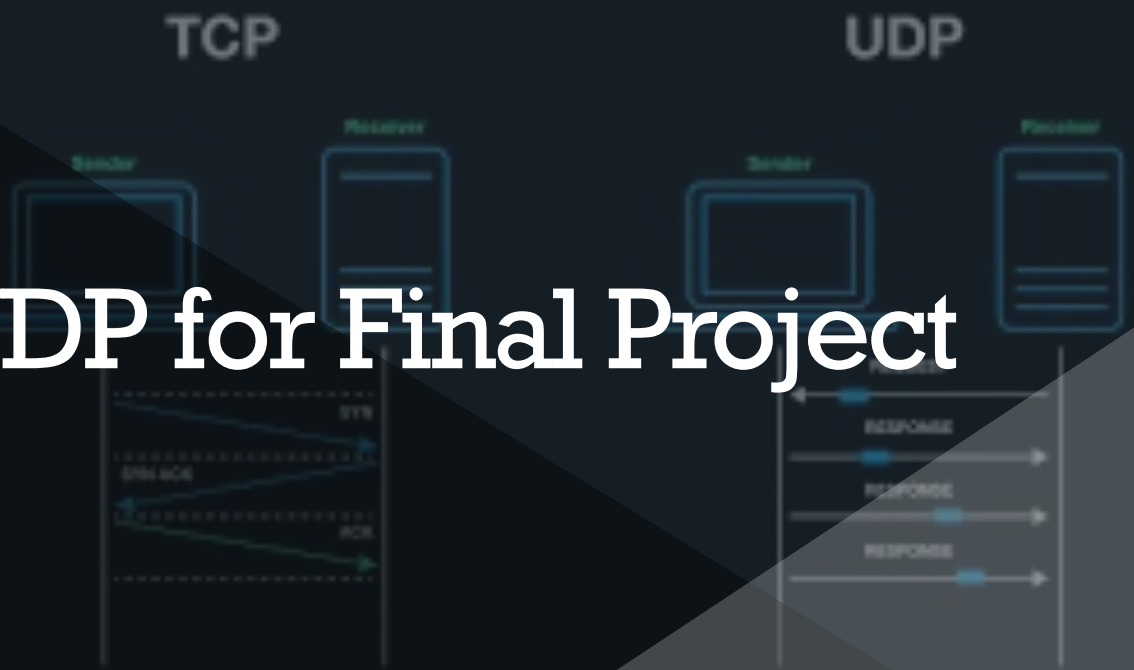
Real time demonstration using Vive510



- Combine with HTML510 to display at 60Hz,with occasional ESP32 lag.
- With two diodes, you can determine orientation in addition to position

03

Using UDP for Final Project



UDP Review

Setup UDP

Global

```
#include <WiFiUdp.h>
WiFiUDP  servername;
```

Goes in setup

```
servername.begin(uint16_t UDP_PORT#);
```

Send a message (buffer)

Send subroutine

```
servername.beginPacket(ipTarget, UDP_PORT#);
servername.write(uint8_t *buffer, length);
servername.endPacket();
```

Receive a message into (buffer)

Receive subroutine (periodically call in loop())

```
int cb = servername.parsePacket();
if (cb>0) servername.read(buffer, length);
```

UDP Messages: GTA 2021C protocol

- IP Addresses (like people's names)
  - 192.168.1.255 is a special address – broadcast to everyone.
- Ports (like phones, where people carry many phones)
  - Port numbers range from 1 to 65535
  - Some port are reserved (typ 1-1023 common preassigned) (1024-49151 are registered with IANA) (49152-65535 dynamic)

`https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`

- Game Messages have 3 parts: #:####,####
  - ID one digit followed by :
  - X value four digits followed by ,
  - Y value four digits
  - Cans send on port 1510,
  - Robots send on port 2510

UDP GTA2021C sender example [part 1 / 2]

```
#include <WiFi.h>
#include <WiFiUdp.h>

const char* ssid     = "TP-Link_05AF";
const char* password = "47543454";

WiFiUDP UDPTestServer;
unsigned int UDPPort = 2510; // port for cans is 1510, port for robots is 2510

// IP ending in 255 is a broadcast address to everyone at 192.168.1.xxx
IPAddress ipTarget(192, 168, 1, 255);
IPAddress ipLocal(192, 168, 1, 10);  // replace with your IP address

void fncUdpSend(char *datastr, int len) {
  UDPTestServer.beginPacket(                    );
  UDPTestServer.write(                          );
  UDPTestServer.endPacket();
}
```

Q8) What parameters would pass for these commands?

On canvas resources 2510-UDPsender.ino

UDP robot sender example [part 2/2]

```
void setup() {
  Serial.begin(115200);
  Serial.print("Connecting to ");  Serial.println(ssid);
  WiFi.config (ipLocal, IPAddress(192, 168, 1, 1), IPAddress(255, 255, 255, 0));
  WiFi.begin (ssid, password);

  UDPTestServer.begin(UDPPort);
}

void loop() {
  char s[13];
  int x, y; // some data, like  xy position
  // store into a string with format #:####,####, which is robotid, x, y
  sprintf(s,"%1d:%4d,%4d",4, x, y);
  fncUdpSend(s,13);
  Serial.printf("sending data: %s",s);
  delay(100);
}
```

Convert two X,Y integers into a char array to fit protocol

On canvas resources 2510-UDPsender.ino

UDP Receiver example [part 1 / 2]

```
#include <WiFi.h>
#include <WiFiUdp.h>
const char* ssid     = "TP-Link_05AF";
const char* password = "47543454";

WiFiUDP canUDPServer;
WiFiUDP robotUDPServer;
IPAddress myIPaddress(192, 168, 1, 9); // change to your IP

void setup() {
  Serial.begin(115200); Serial.print("Connecting to ");  Serial.println(ssid);
  WiFi.config(myIPaddress, IPAddress(192, 168, 1, 1), IPAddress(255, 255, 255, 0));
  WiFi.begin(ssid, password);
  canUDPServer.begin(1510); // can UDP port 1510
  robotUDPServer.begin(2510); // robot UDP port 2510
}

void loop() {
   handleCanMsg();  // periodically call handlers
   handleRobotMsg();
   delay(10);
}
```

Two UDP servers, one for each port number

On canvas resources game-UDPreceiver.ino

UDP Receiver example [part 2/2]

```
void handleCanMsg() {
  const int UDP_PACKET_SIZE = 14; // can be up to 65535
  uint8_t packetBuffer[UDP_PACKET_SIZE];

  int cb = canUDPServer.parsePacket(); // if there is no message cb=0
  if (cb) {
    int x,y;
    packetBuffer[cb]=0; // null terminate string

    canUDPServer.read(packetBuffer, UDP_PACKET_SIZE);

    x = atoi((char *)packetBuffer+2); // #,####,#### 2nd indexed char
    y = atoi((char *)packetBuffer+7); // #,####,#### 7th indexed char

    Serial.print("From Can ");
    Serial.println((char *)packetBuffer);
    Serial.println(x);
    Serial.println(y);
  }
}
```

Convert char array string to an integer

Q9 How would handleRobotMsg() be different?

Note about char and uint8_t

- Why are we casting (char *) or (uint8_t *) sometimes and not others?

```
uint8_t packetBuffer[UDP_PACKET_SIZE];
canUDPServer.read(packetBuffer, UDP_PACKET_SIZE);
x = atoi((char *)packetBuffer+2); // #,####,#### 2nd indexed char
y = atoi((char *)packetBuffer+7); // #,####,#### 7th indexed char


void fncUdpSend(char *datastr, int len) {
  UDPTestServer.write((uint8_t *)datastr, len);
}
```

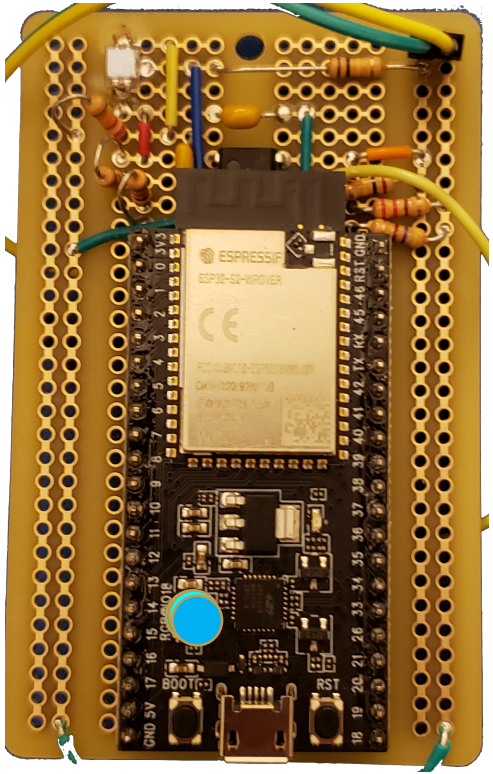
- For ASCII text strings, it doesn't make a difference if we use char or uint8_t. But in general, uint8_t refers to data that can be any binary number, and char refers to text.
- Here we are using generic routines that can send binary to send text.

Vive/UDP Test unit

- Start up trying to connect to GM router TP-link_05AF – blinks red and green.

- If locked on to Vive turns blue-green, transmits xy location for can#1
  - E.g. "1:1234,5436"
- If sync is lost (or diode is blocked then turns red)

- If no Vive light house detected blinks blue (lighthouse may be unplugged)



String manipulations

- We use ascii strings e.g., "1:1235,1234" for our GTA protocol. We could use binary to represent numbers.
  - `itoa()` converts integer to alpha, e.g., 1234 -> '1','2','3','4'
  - `atoi()` converts alpha to integer e.g., four chars "1234" -> 1234

- Arduino ESP32 has a String type. This uses malloc() and free() to dynamically allocate memory space… which works for short term things, but is generally not a good idea for production embedded systems.
  - HTML510.cpp uses the String type… I should probably change this.

Summary

- Vive tracker uses timing for decentralized localization
- Photodiodes are required to handle fast signals
- UDP messages can be broadcast to all on a subnet using 255 as the last number in the IP address
- UDP ports can be used like phone numbers to reach the same people but via a different channel.
- You need your robot to broadcast XY location
- You need to receive XY locations of cans

Answer in CHAT

Answer how you feel about each topic below with:

1. I don't understand this topic at all
2. I don't know now, but know what to do to get by
3. I understand some, but expect to get the rest later
4. I understand completely already

A. How Vive works

B. Building vive circuit and using Vive510 software

C. UDP for robot and can messaging with game protocol.