Lecture 24

More on Communication Examples for Scanning

Agenda

- Master-Slave / Client-Server Communications (AJAX)
- Distance Ranging
- Extra bits

Requirements Rubric

Up to 3 minutes per task

• Controlled Mobile Robot	50pt controlled to arbitrary point
• One can in doubling square	10pt
• Follow wall – full circuit	10pt partial credit: % of full circuit
• Light tracking – across ½ field	10pt partial credit: % of 1/2 field
• Steal two opponent cans	15pt per can up to 2 cans
• Total possible	110pts (10 pts extra credit)

Zoom Meeting

Cancel the Spotlight Video

Game Field View

Canvas->files->resources->
how to make a can.pdf

Final Project Tentative Schedule

April	4	5	**6**	7	8	9 **Design Review 1**	10
	11	12	13	14	15	16	17
	18	19 **Today**	20	21	22	23 **Design Review 2**	24
	25	26 **Last day for laser and 3Dprint**	27	28 Last Class	29 **Report due date**	30 Reading days	1 Reading days
May	2 Reaeding days	3 Reading days	4 **Grading Evaluation**	5 **Grading Evaluation**	6 **Grading Evaluation**	7 **Grading Evaluation**	8 **Day 1 Group Stage**
	9 Day 2 Group Stage	10 Final Brackets	11 Last day Report accepted	12	14	15	15

01

Master-slave Client-server Communication

I2C Master – Slave communications

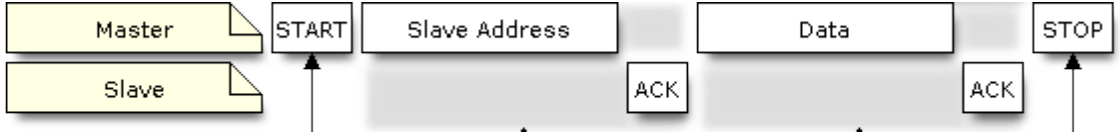
WRITE SEQUENCE

M: Start / Address (write)

S: Ack (0)

M: Write Data

S: Ack (0)



READ SEQUENCE

M: Start Address (read)

S: Ack (0)

S: Read Data

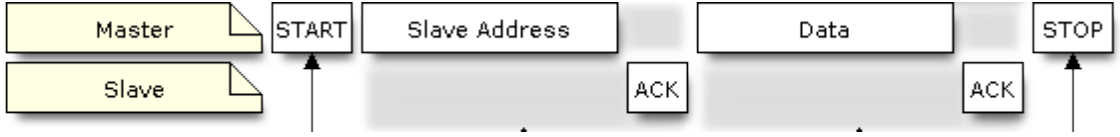
M: Nak (1)

I2C Master – Slave communications

WRITE SEQUENCE

Only master can initiate transfer.

Me: Joe!	• Address (write)
Joe: Yes?	• Ack (0)
Me: Think about resistor color for 2.	• Write Data
Joe: Ok	• Ack (0)



READ SEQUENCE

Me: Joe?	• Start / Address (read)
Joe: Yes?	• Ack (0)
Joe: Red	• Read Data
Me: Thanks	• Nak (1)

SPI Master – Slave communications

Only master can initiate transfer.

Me: Joe

- SS line activated

Me: What is resistor color for 2.

- Write Data

Me: Joe

- SS line activated

Me: What is resistor color for 3

- Write and Read Data

Joe: Red

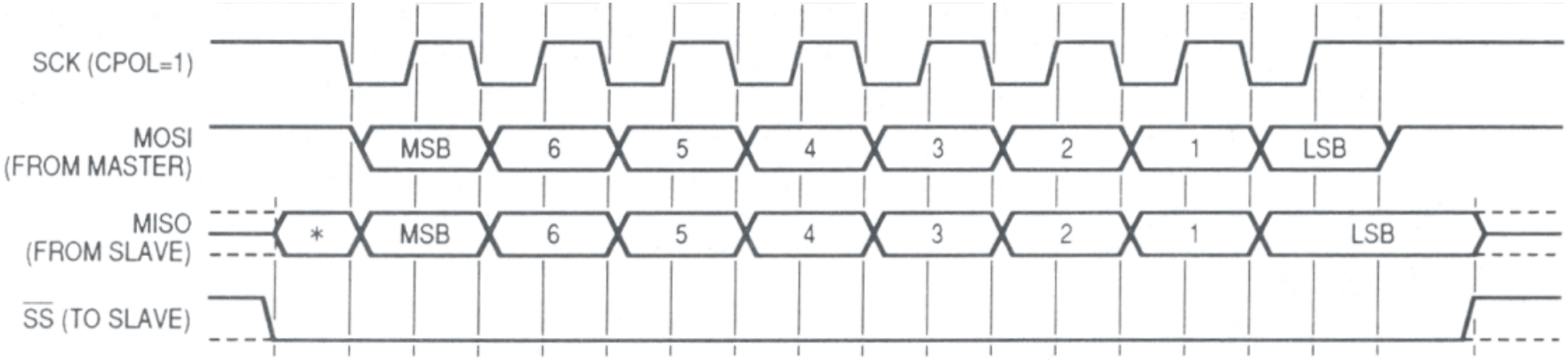
Me: Joe

- SS line activated

Me: What is resistor color for 4

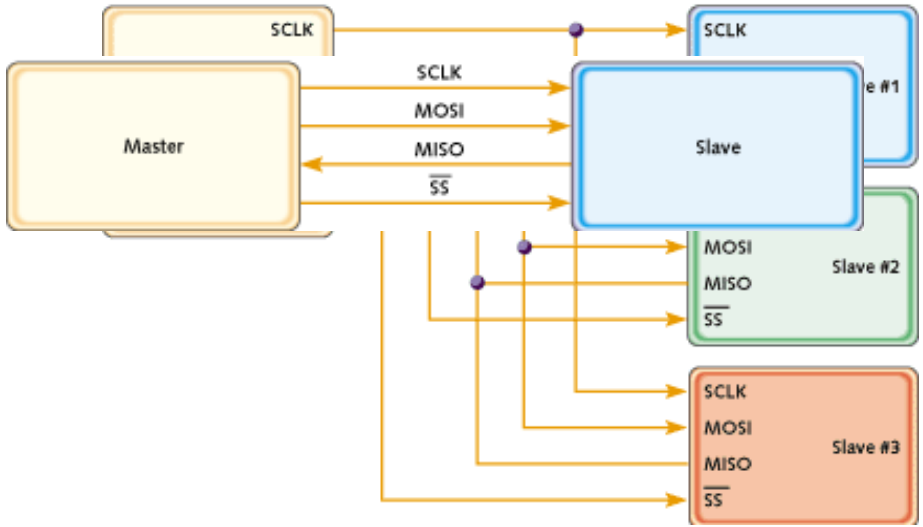
- Write and Read Data

Joe: Orange



Serial Peripheral Interface (SPI)

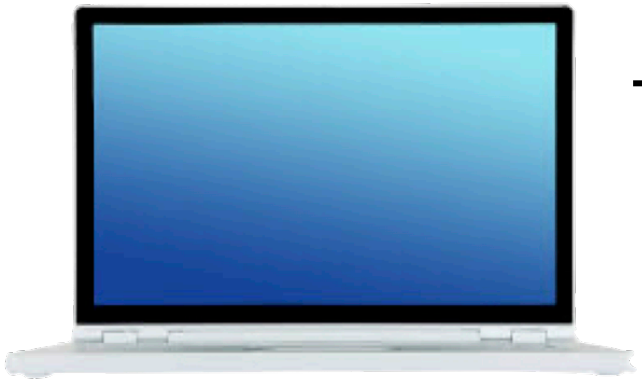
- Multidrop: need one SS line for each slave
- Lower overhead than I2C (no address, no R/W bit, full duplex)
- # of slaves limited by # of SS lines (digital output pins)
- I2C can have up to127 slaves
  - Up to bus capacitance 400pf
- Cannot have multi-master.



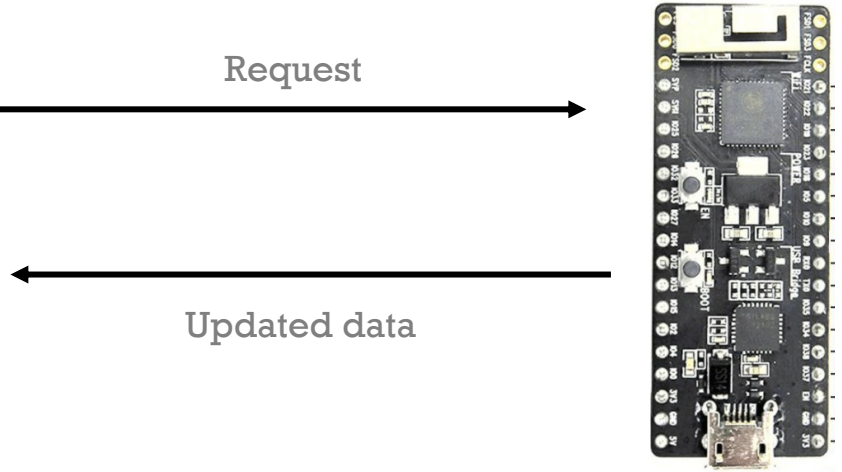
Web sites have a Client-Server Model (like master-slave for comms)

Client website
Initiates communication

Server code
Responds to client



Request



Updated data

Cannot send data without request from client

Updating info on website

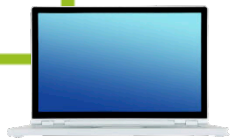
- How does Oscillosorta update the display data being constantly updated by ESP32?
- AJAX – Asynchronous JavaScript And XML.
  - Read data from a web server - after the page has loaded
  - Update a web page without reloading the page
  - Send data to a web server - in the background
  - https://www.w3schools.com/js/js_ajax_intro.asp
- Two mechanisms
  - Timed calls setTimout(), setInterval()
  - XMLHttpRequest objects

AJAX

Browser

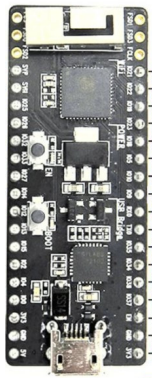
An event occurs...

- Create an XMLHttpRequest object
- Send HttpRequest



Request for display info

Server



Display info

https://www.w3schools.com/js/js_ajax_intro.asp

AJAX XMLHttp

REQUEST

```
function getData() {
  var xhttp = new XMLHttpRequest();

  xhttp.open("GET", "up", true);
  xhttp.send();
}
```

Handler URL text

RESPONSE

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200 ) {
    // draw new data in dark green
    ch = this.responseText.split(","); // get ranging data
    ctx.strokeStyle = "#008800";
    drawDataCircles();
  }
};
```

AJAX XMLHttp template

```
function getData() {
  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200 ) {
      callbackfunction ();
    }
  };

  xhttp.open("GET", URLtext, true);
  xhttp.send();
}
```

AJAX Server Polling

- Use `setInterval()` to periodically request server data

```
setInterval( function, interval_in_ms);
setInterval(getData, 1000);
```

getData called once every 1000ms.

- Use `setTimeout()` inside a called function, timing based on function run time

```
function getData() {
  ...
  ...
  setTimeout(getData, 100);
}
```

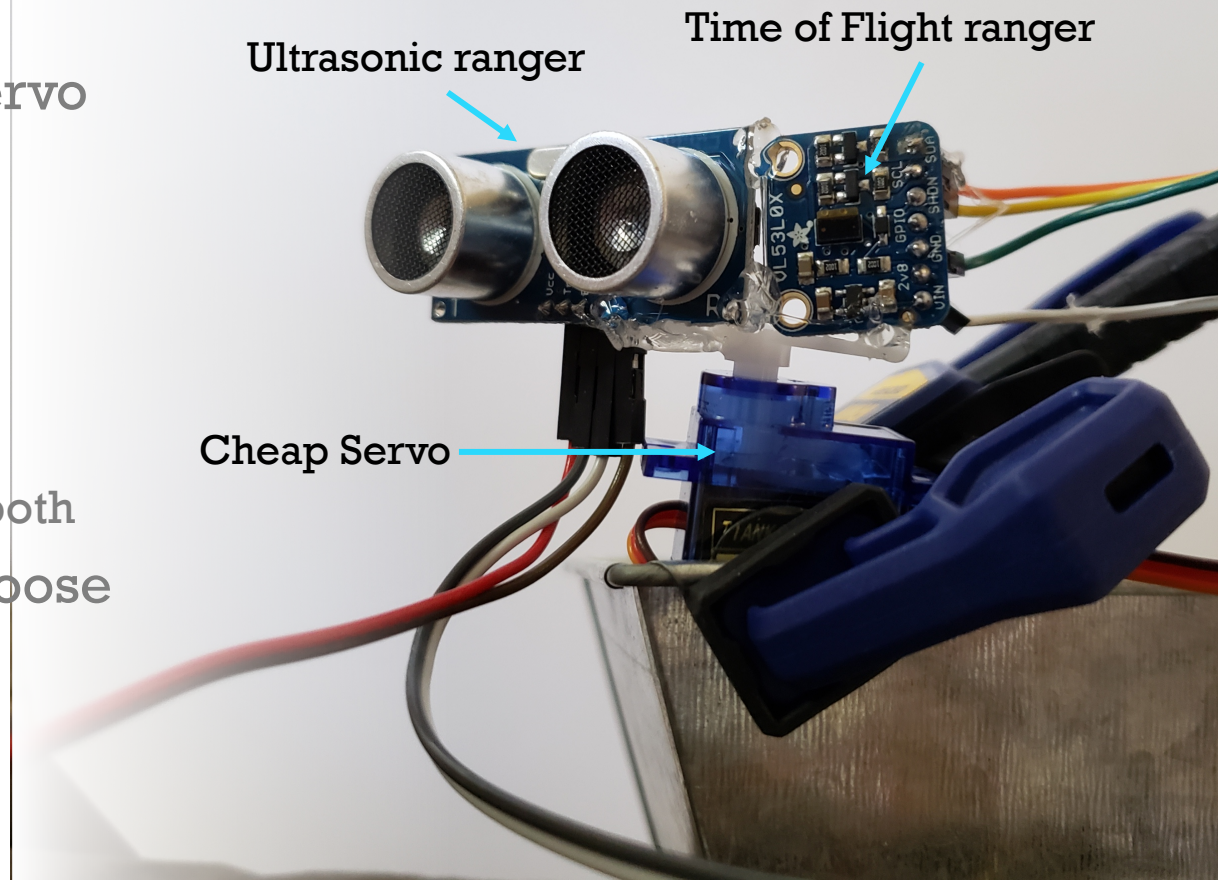
getData called 100mS after the function runs

02

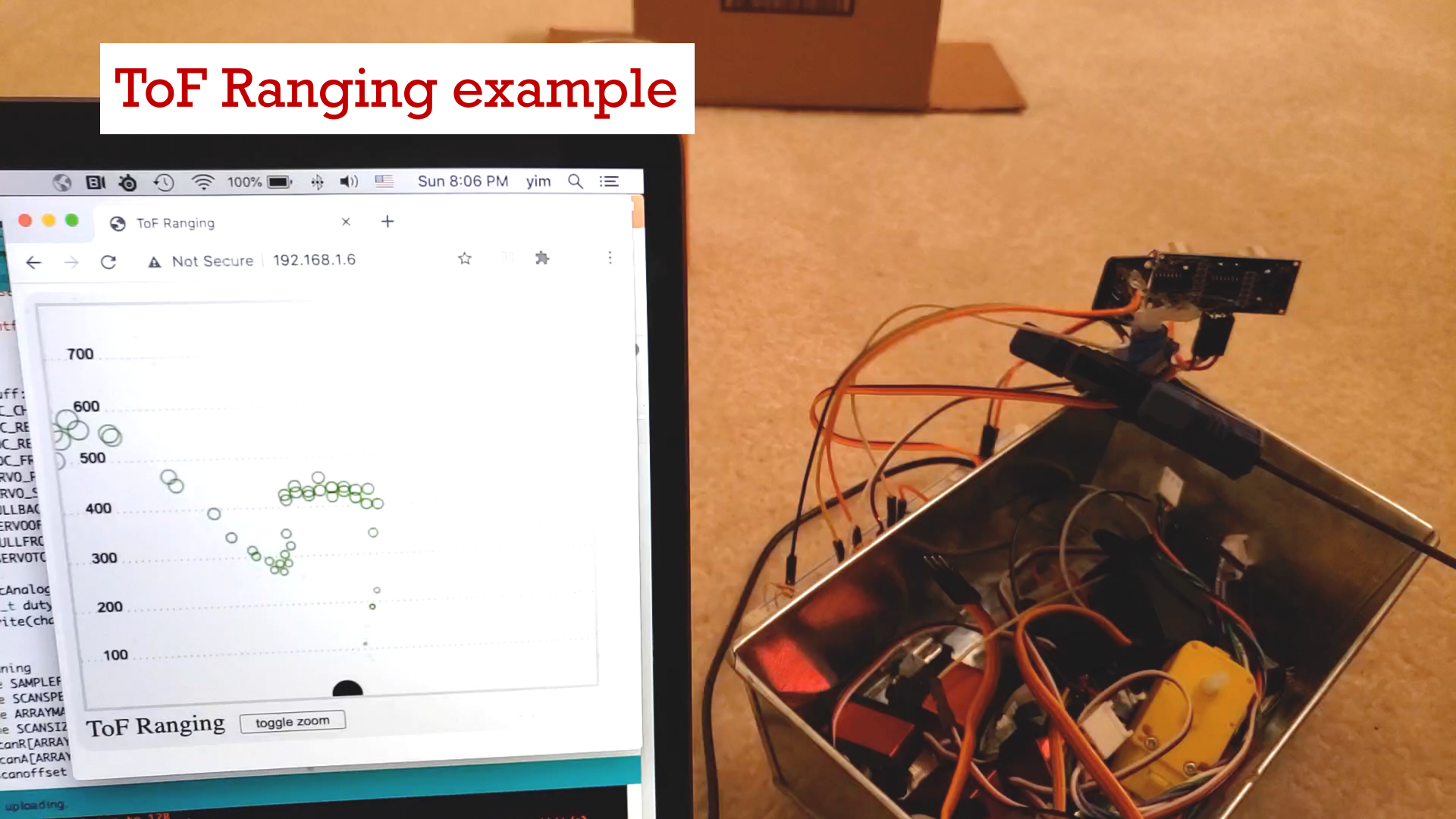
Distance Ranging example

Ranging Example

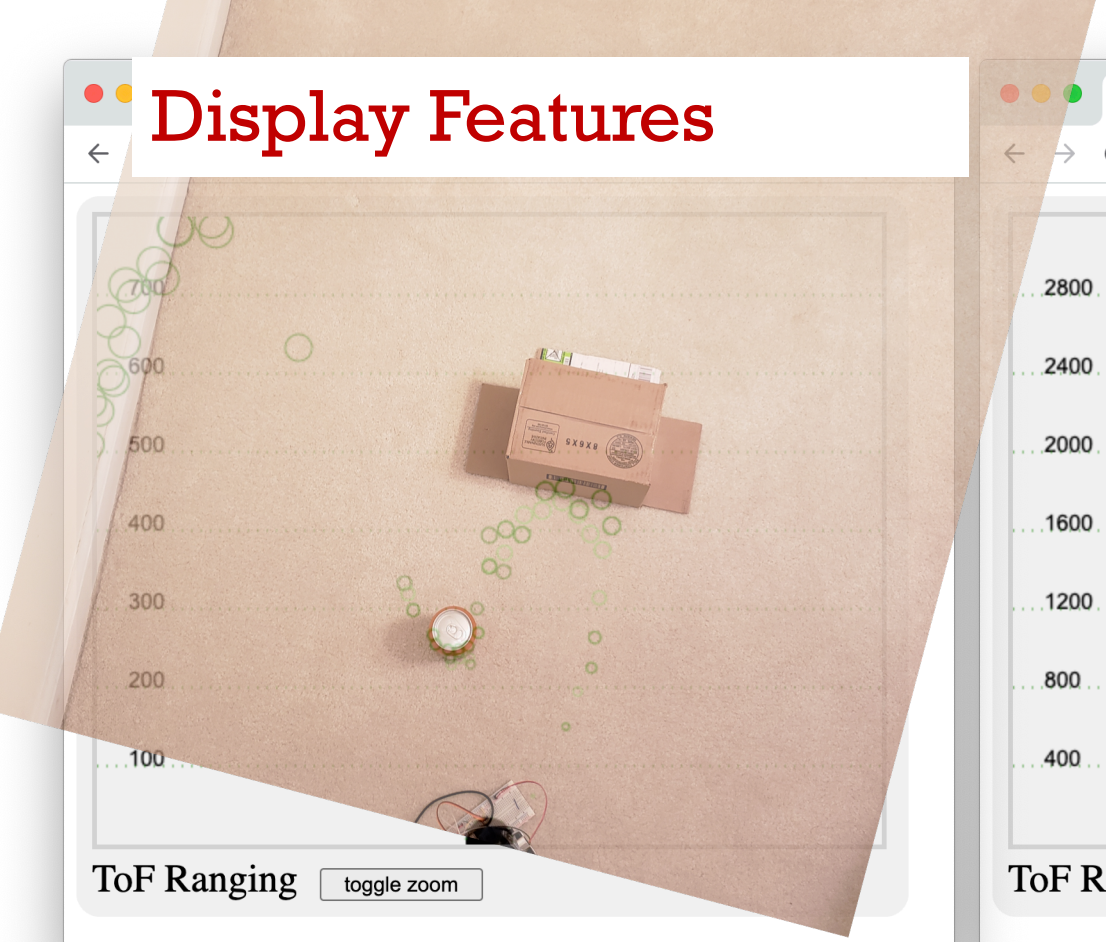
- Rangers hotglued to servo horn
- Servo clamped to box
- ESP32
  - 2 GPIO for ultrasonic
  - I2C for ranger
  - Power and ground for both
- Code runs set up to choose one sensor.



ToF Ranging example



Display Features

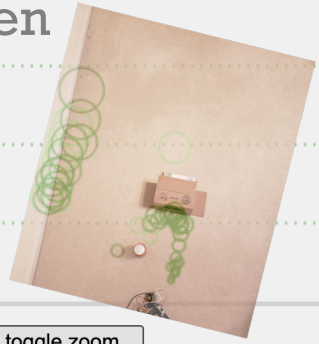


ToF Ranging

Not Secure | 192.168.1.6

- Distance in mm
- Circles denote resolution
- Zoom button
- Past data shown in light green

2800
2400
2000
1600
1200
800
400



ToF Ranging toggle zoom

Ranging Example has 5 parts

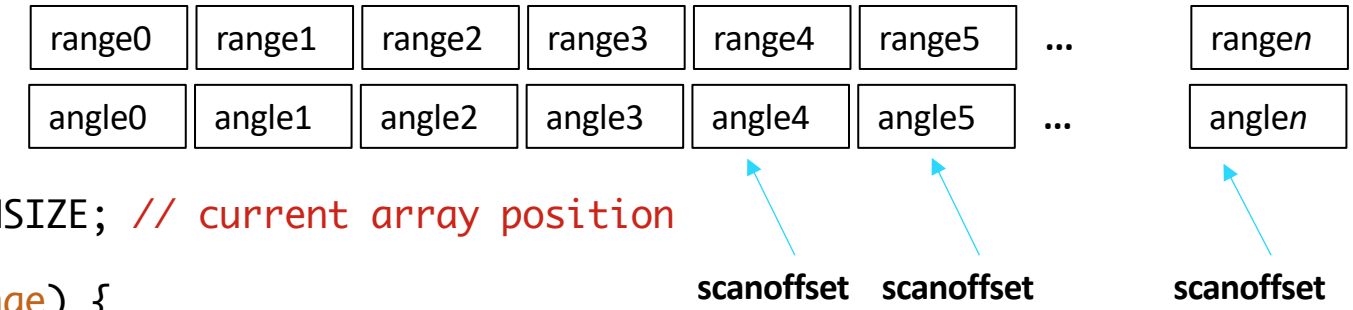
- ✓ Range Sensor Reading
- ✓ Servo Commands
- • Scanning and storing
- ✓ Web communication
- • Javascript Display

Code Structure

```
void setup() {
  ...// normal setup for serial, web, sensors, servo
  attachHandler("/up",handleUpdate);
  attachHandler("/favicon.ico",handleFavicon);
  attachHandler("/ ",handleRoot);
}

void loop() {
  static uint32_t lastServo = micros(), lastmicros = micros(), us = micros();

  us = micros();
  if (us-lastmicros > 5000){              // check for webpage request
    lastmicros = us;
    serve(server,body);
  }
  if (us-lastServo > 1000000/SAMPLEFREQ) { // update the servo position
    scanStep(rangeToF());  // can use rangeSonar() for ultrasonic
    lastServo = us;
  }
}
```

Scanning and storing

```
int scanR[ARRAYMAX];
int scanA[ARRAYMAX];

int scanoffset = SCANSIZE; // current array position

void scanStep(int range) {
  static int angle;
  static int dir=SCANSPEED;

  if (angle+SERVOOFF > FULLFRONT)  dir = -SCANSPEED;
  if (angle+SERVOOFF < FULLBACK)   dir = SCANSPEED;

  scanR[scanoffset % ARRAYMAX] = range;
  scanA[scanoffset % ARRAYMAX] = -SERVOTODEG(angle);
  scanoffset++;
  angle += dir;
  ledcAnalogWrite(LEDC_CHANNEL, SERVOOFF+angle, LEDC_RESOLUTION);
}
```



Why store values in array?
No storage method: send each point as you get it

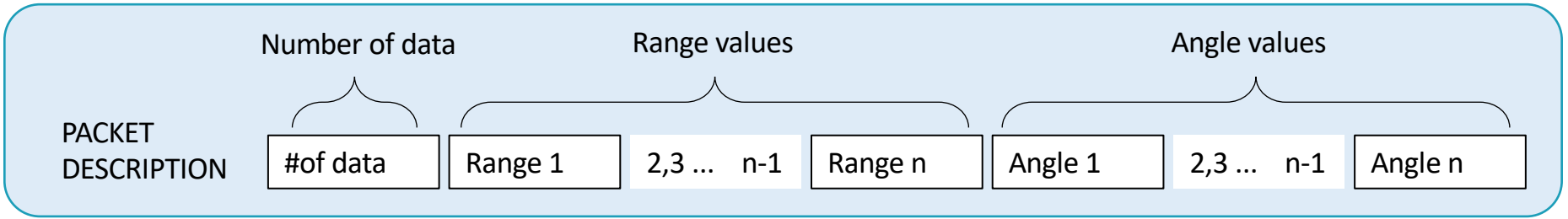
- Can't push the data out to the website
- Also overhead is too large. Each HTTP request results in ~1000 extra characters.
- GTA 2021 rules: 10k byte/sec transmission limit
  - -> ten packets max per second.

```
GET /L HTTP/1.1
Host: 777f4a18c8ef.ngrok.io
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Referer: http://777f4a18c8ef.ngrok.io/H
Upgrade-Insecure-Requests: 1
X-Forwarded-For: 172.58.207.29
X-Forwarded-Proto: http
```

```
GET /favicon.ico HTTP/1.1
Host: 777f4a18c8ef.ngrok.io
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Referer: http://777f4a18c8ef.ngrok.io/L
X-Forwarded-For: 172.58.207.29
X-Forwarded-Proto: http
```

Server Sending Data

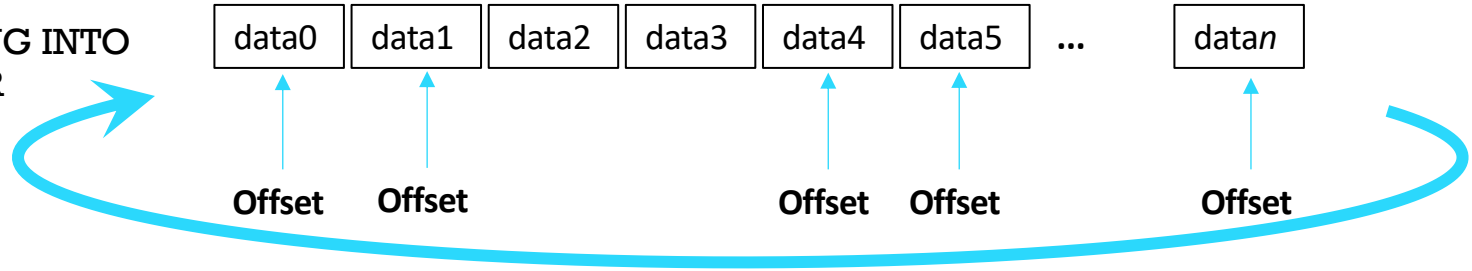
```
void handleUpdate() {
  String s = "";

  s = s+SCANSIZE; // number of data pairs
  for (int i=0; i<SCANSIZE; i++) { //  range value
    s = s+","+ scanR[(scanoffset-i) % ARRAYMAX]; // range lags angle by 1
  }
  for (int i=0; i<SCANSIZE; i++) { //  angle value
    s = s+","+ scanA[(scanoffset-i-1) % ARRAYMAX];
  }

  sendplain(s);
}
```



Ring Buffer (AKA Circular Buffer)

```
scanR[scanoffset % ARRAYMAX] = range;
```

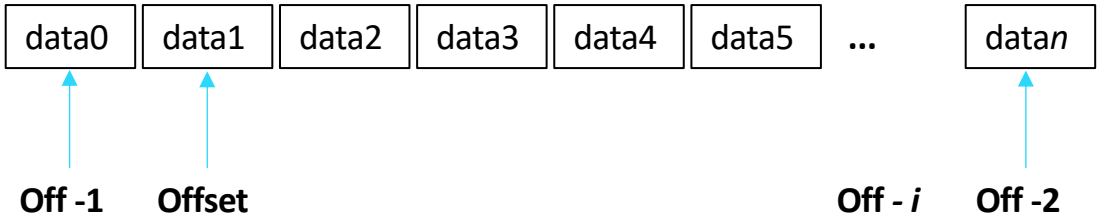
STORING INTO BUFFER



Note: reading and storing operations are asynchronous

```
s = s+","+ scanR[(scanoffset-i) % ARRAYMAX];
```

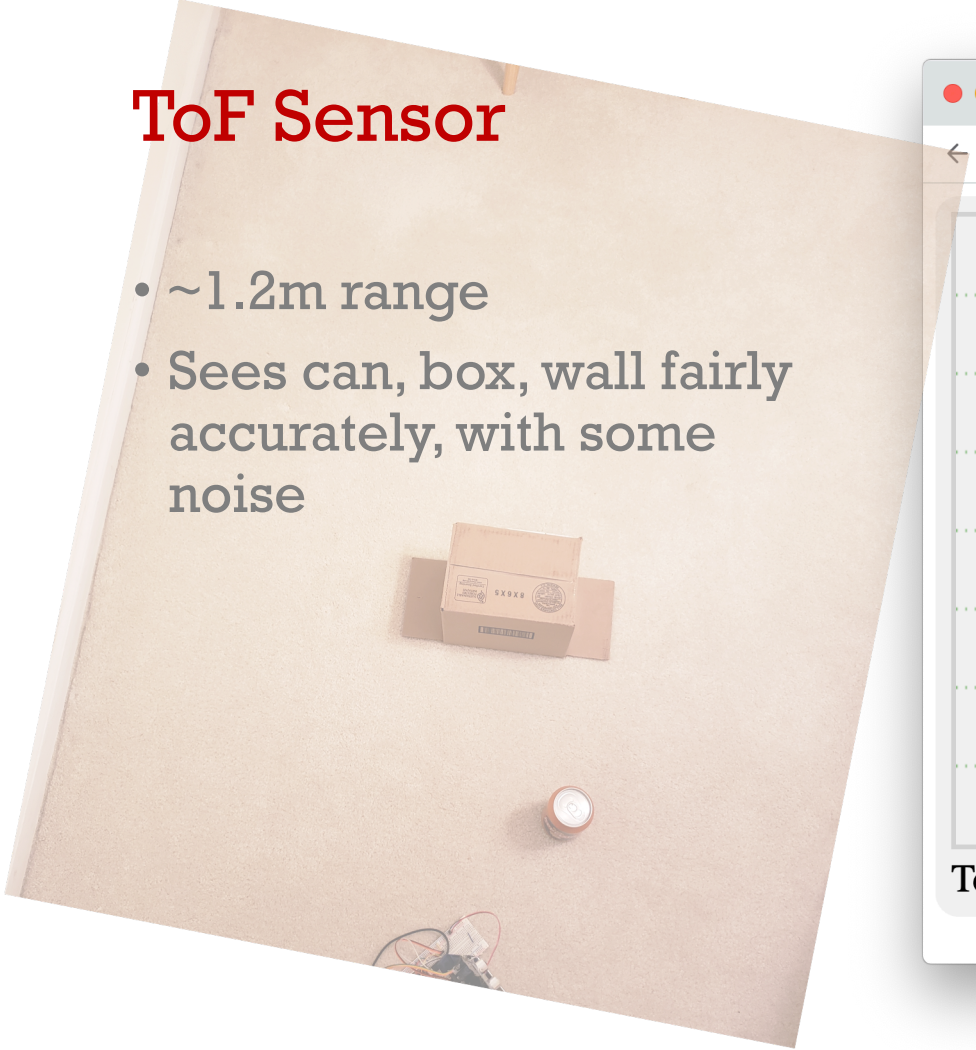
READING OUT OF BUFFER

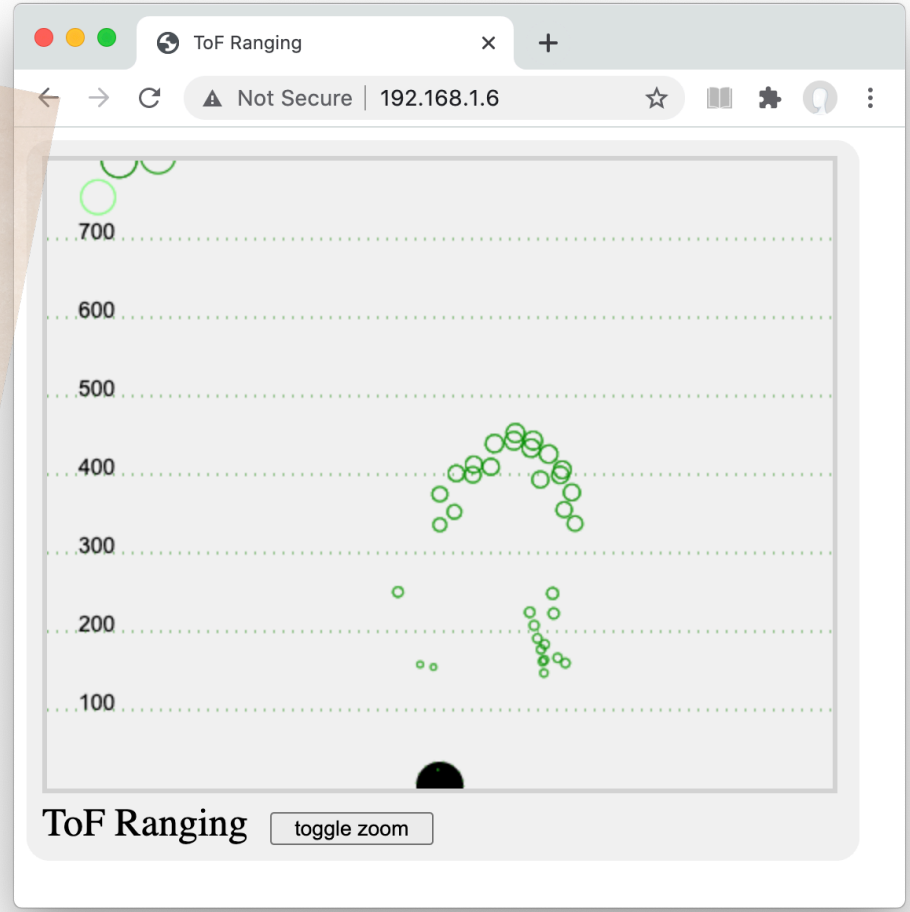


Want most recent group of data (start at current offset)

ToF Sensor

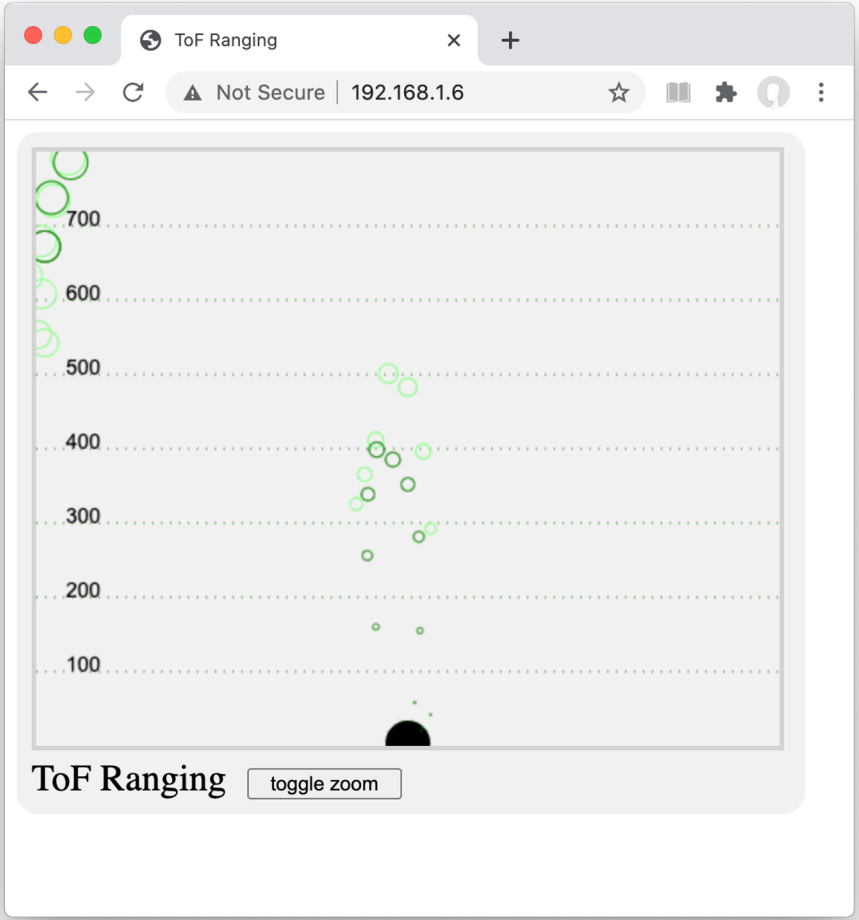
- ~1.2m range
- Sees can, box, wall fairly accurately, with some noise





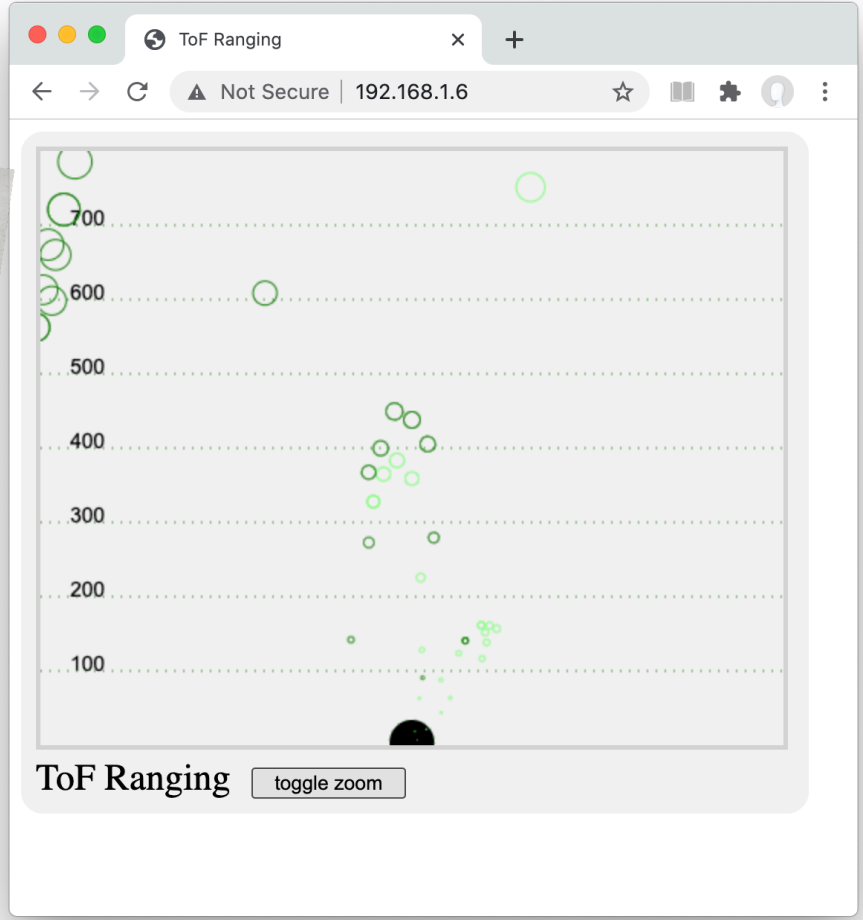
ToF

- Is it seeing the box?



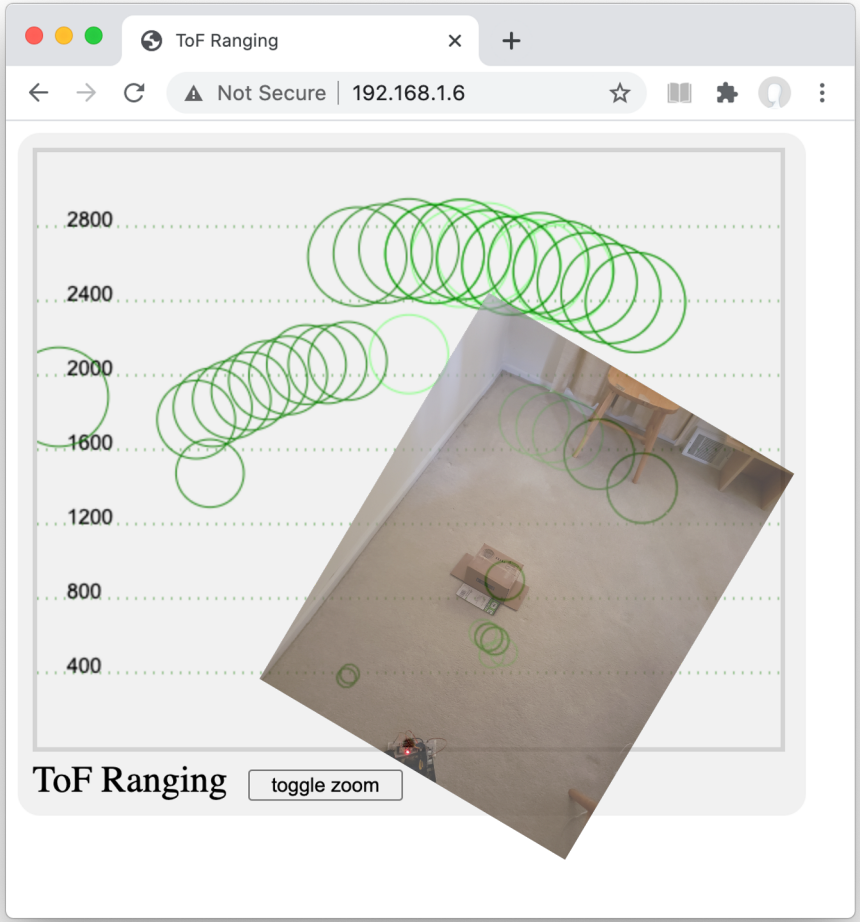
ToF

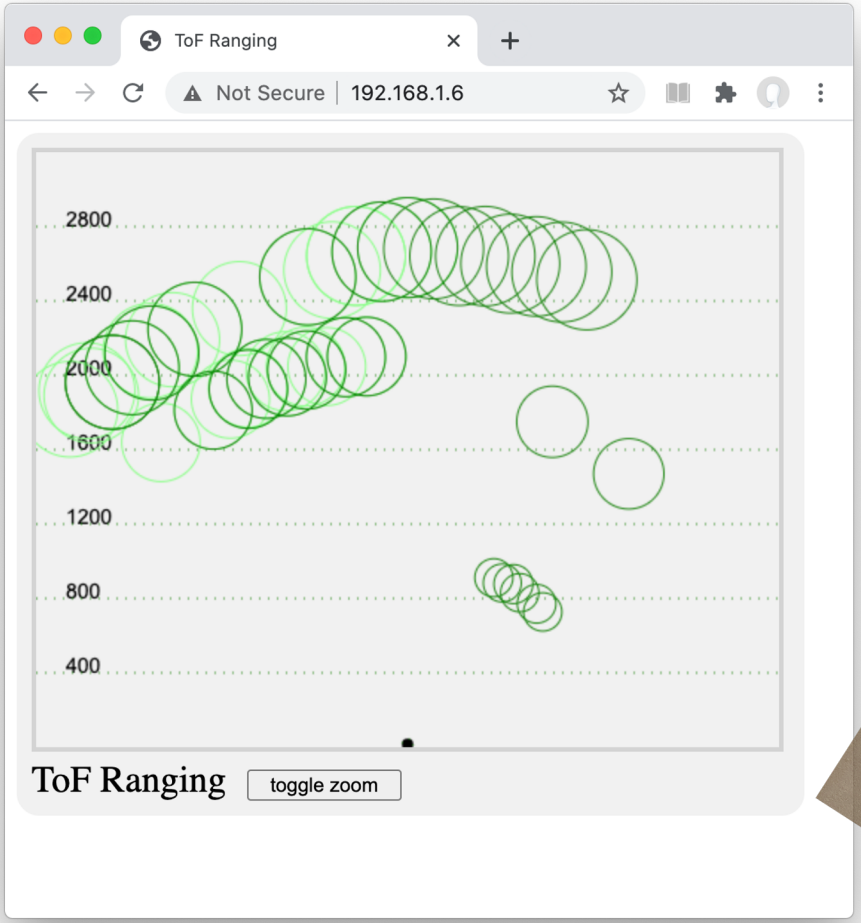
- Phantom data
- Seeing the floor
- Need to angle up slightly



Ultrasonic

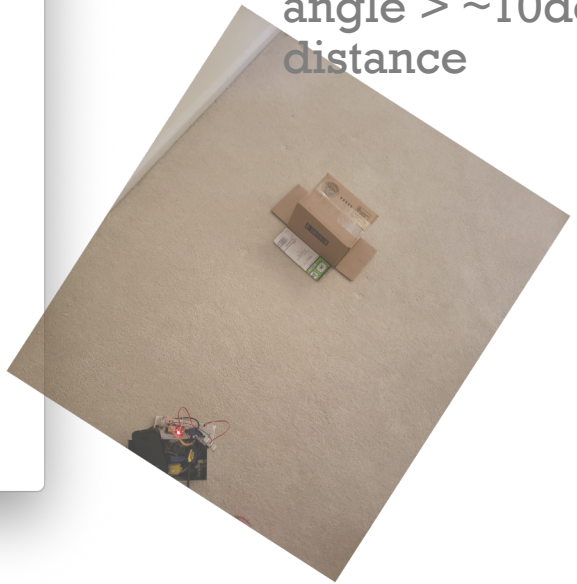
- Note longer range.
- Random phantom points
- Wall ranges are distorted
- Specular reflection helps (doesn't see floor)

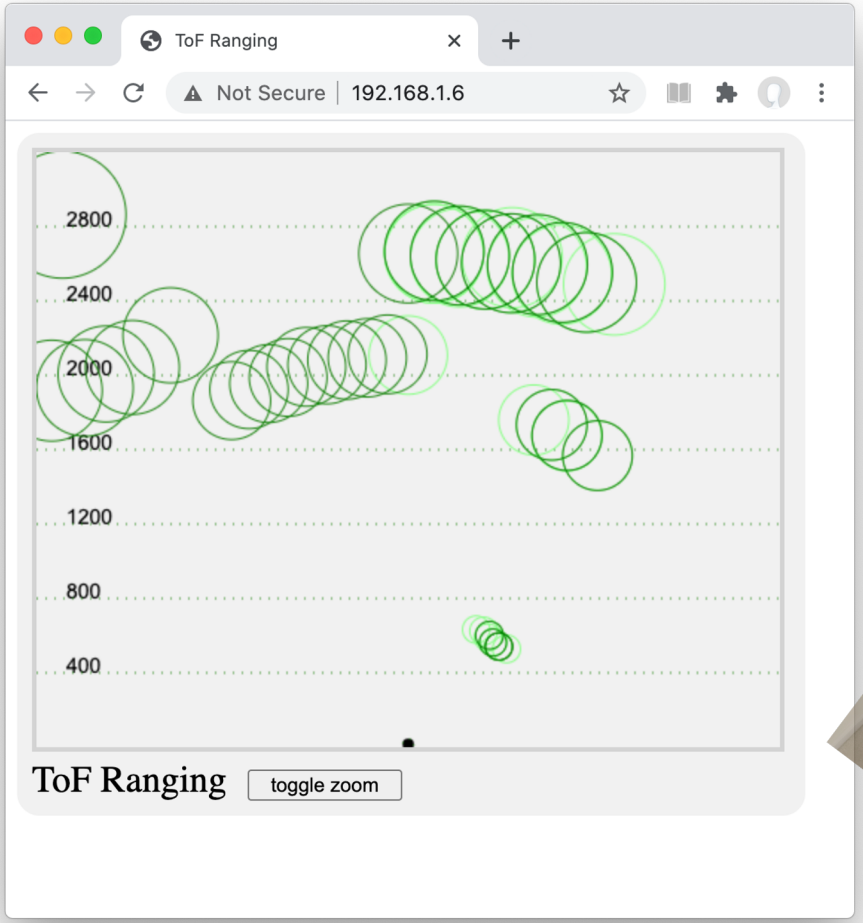




Ultrasonic

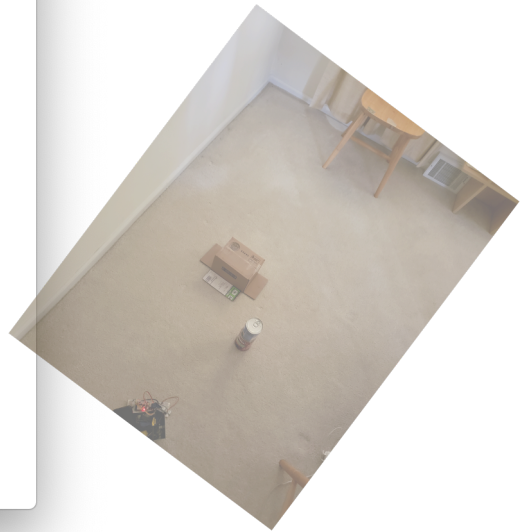
- Sees box clearly when perpendicular to sensor.
- Specularity could be a problem
  - E.g. when flat surfaces are at an angle > ~10deg dending on distance

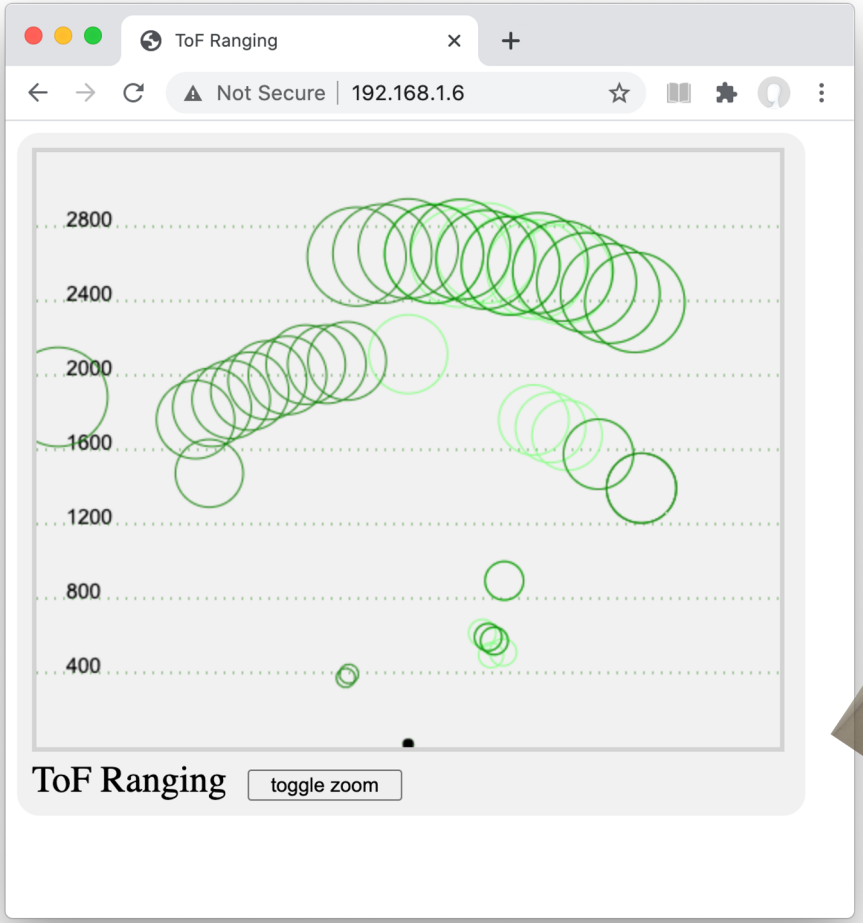




Ultrasonic

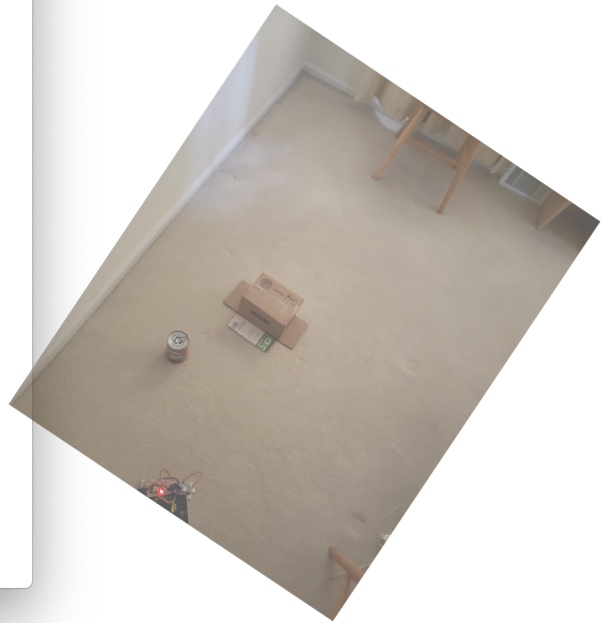
- Can't see the can





Ultrasonic

- Maybe sees can, maybe noise…



Quiz on scanning

For each question, in private chat answer: TOF, Ultrasonic, Both or Neither.

Q1: Which system has higher precision?

Q2: Which is faster?

Q3: Which costs about 4x's more?

Q4: Which would you use to see a small object on the floor?

Q5: Which has longer range?

Q6: Which cannot see soda cans reliably

Q7: Which is not likely to see the 2x4 field walls?

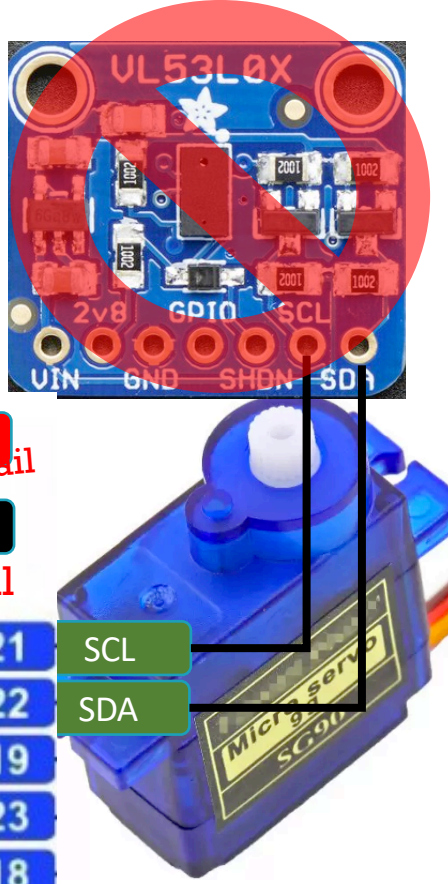
03

Debugging Story

ToF Scanning Demo Development

I miss-wired here so it works

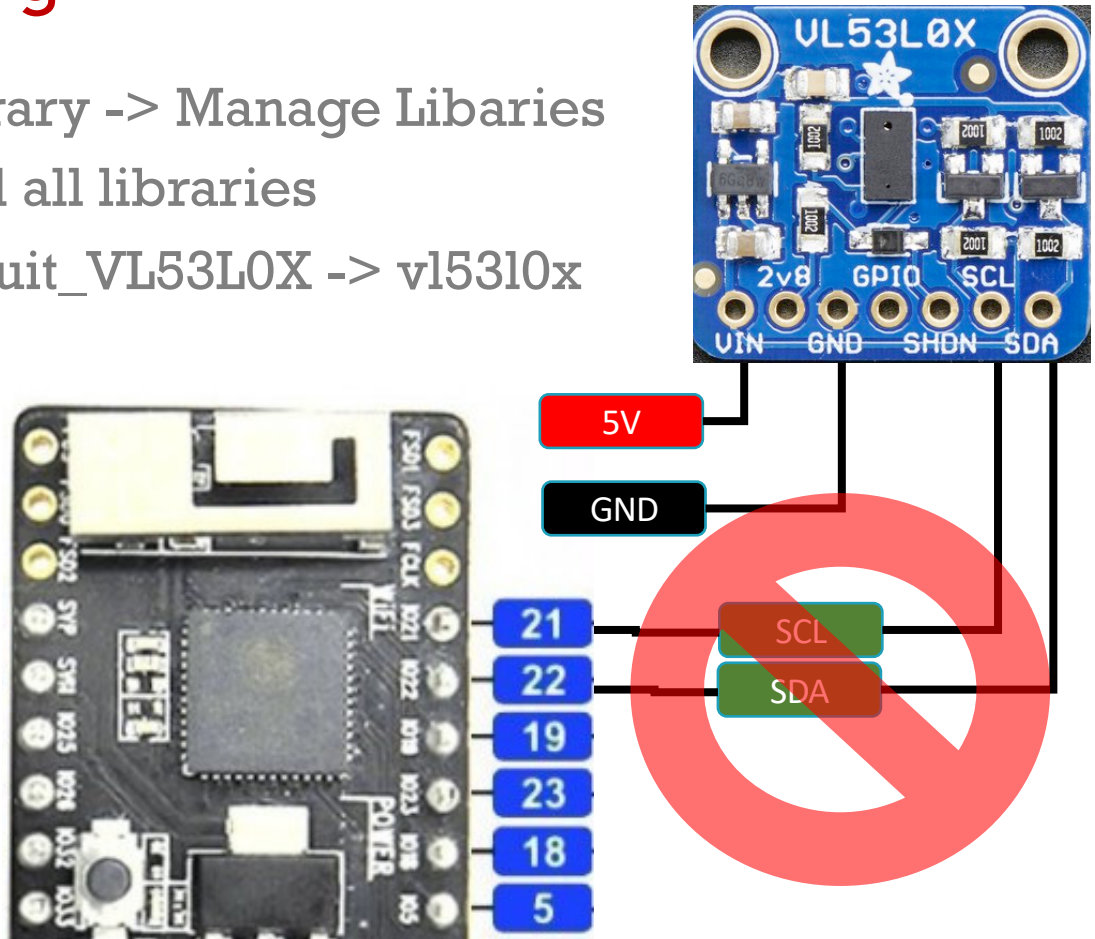
- Test VL53L0X using I2C and sample code on ESP32 - success
- Write slides for L23 L23 wiring is WRONG
- Add servo and scanning for this lecture
- Mount VL53L0X to servo and test - success
- Servo motion pulls wires out – rewire – fail
- Test wiring (recheck wiring with slides) (Suspect wire connection) – fail
- Test with previous VL53L0X sample code (Suspect software) – fail
- Test new ESP32 (Suspect ESP32 ports) – fail
- Check VL53L0X voltages (Suspect VL53L0X board failure) - OK
- Switch to ultrasonic sensor for scanning demo
- Try new I2C address (based on datasheet, also rewire new ESP32) – success!???
- Move back to old ESP32 connected to servo – fail
- Realize error in wiring TWICE! (Bad lecture notes is culprit!)



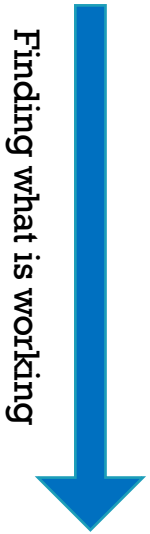
L23 Slide on Wiring

- IDE Sketch -> Include Library -> Manage Libaries
- Search for VL53L0X install all libraries
- File -> Examples -> Adafruit_VL53L0X -> vl53l0x

- ESP32 default SCL is ~~GIPO21~~ GPIO22
- ESP32 default SDA is ~~GPIO22~~ GPIO21
- Attach SDA to SDA
- Attach SCL to SCL



Order of trust (correct and working properly)



1. Corporate documentation
2. Test equipment:
   - E.g. DMM, Power Supply, Oscilloscope
3. Passive parts
   - Resistors, caps, wires
4. Simple active parts
   - LEDs, diodes, voltage regulators
5. Oscillosorta
6. Lecture documentation
7. Web documentation

90% of ERRORS DUE TO THESE:

12. Blown port on MCU
13. External hardware
    - H-bridge, logic, opamps, sensor, motor, driver
14. Wiring
    - breadboard interface, connectors,
15. Software

04

Extra bits

Multi-master (peer)

- Use zoom raise hand feature
- Answer questions on resistor values

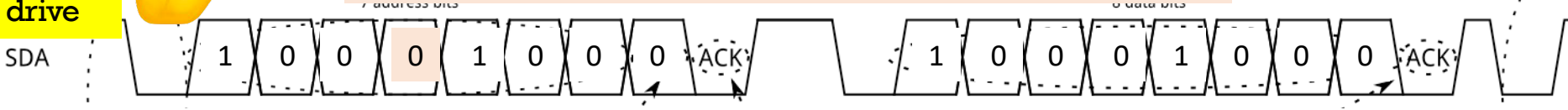
- Students are the multiple-masters. Anyone can talk, but need arbitration to determine who talks first.

- Use zoom chat feature to private chat questions.
- Students are clients. Teacher is server.

I2C multi-master arbitration – relies on OC output "wired-AND"

Master 1 intends to drive

Master 1
Checks to see if the SDA is same doen't see a difference

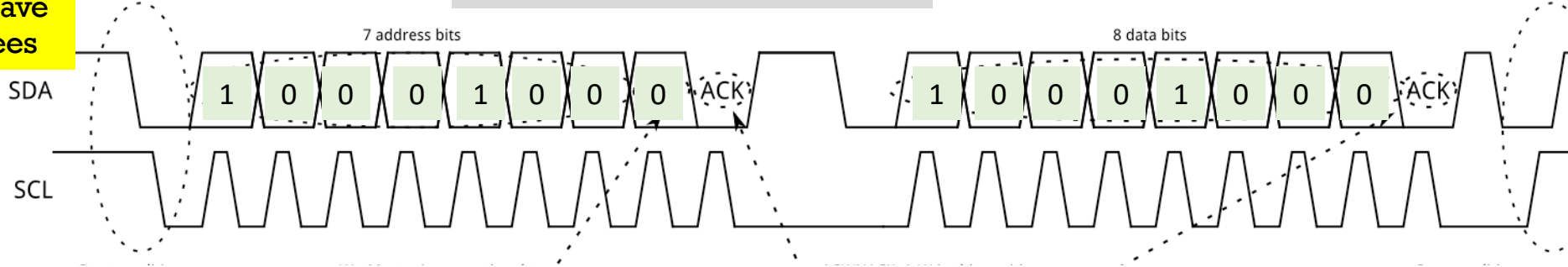


Master 2 Intends to drive

Master 2
Checks to see if the SDA is same

Master 2
Stops driving SDA, will try again

Slave sees

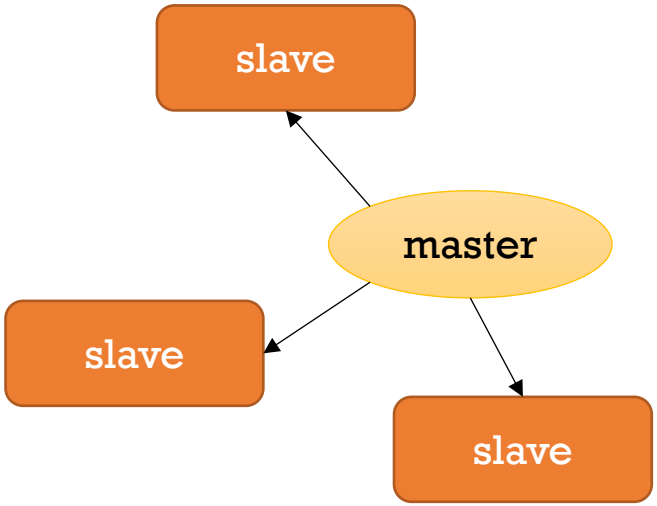


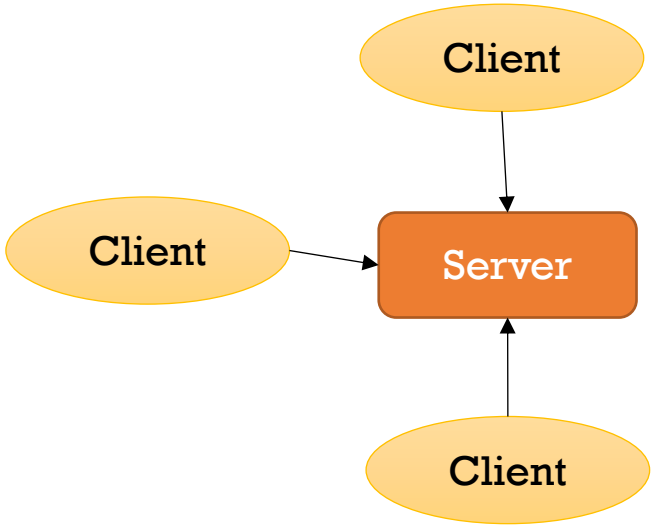
Other arbitration (collision detection)

- SPI – cannot happen, single master only
- Ethernet – random backoff time.
  - Randomly choose one of 1024 time slots
  - If collision occurs back off again (upto 16 times)
  - Impacts usage for real-time (though large variable frame size is the main reason).
- WiFi – RF frequencies avoid transmitting at same time.
  - 802.11 has CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) and RTS/CTS (Request To Send/Clear To Send).
  - Random backoff time if collision expected
  - ESP32 WiFi has limited concurrent clients (maybe 4?)

Difference between Master-Slave and Client-Server

- Mostly same, but typically master-slave is many-slave to one-master. Client-server is many-client to one-server





Aside – on coding (all valid);

Which is better?

A:

```
int steps=1, bumpedflag=FALSE;
```

or

B:

```
int steps=1, bumpedflag;
```

or

C:

```
int steps=1;
bool bumpedflag;
```

Gobal variables are all initialized to 0 (not true for local variables). But explicitly assigning to 0 doesn't hurt and indicates intention

To use `bool` you must `#include <stdbool.h>`

Aside – on coding;

Which is better?

```
TCCR1B = (1<<CS10) | (1<<CS12);
```

Slightly faster and shorter
(smart compiler will join into one command)

vs

```
set(TCCRIB,CS10);
set(TCCRIB,CS12);
```

Is a little clearer,
(won't get confused with | or &)

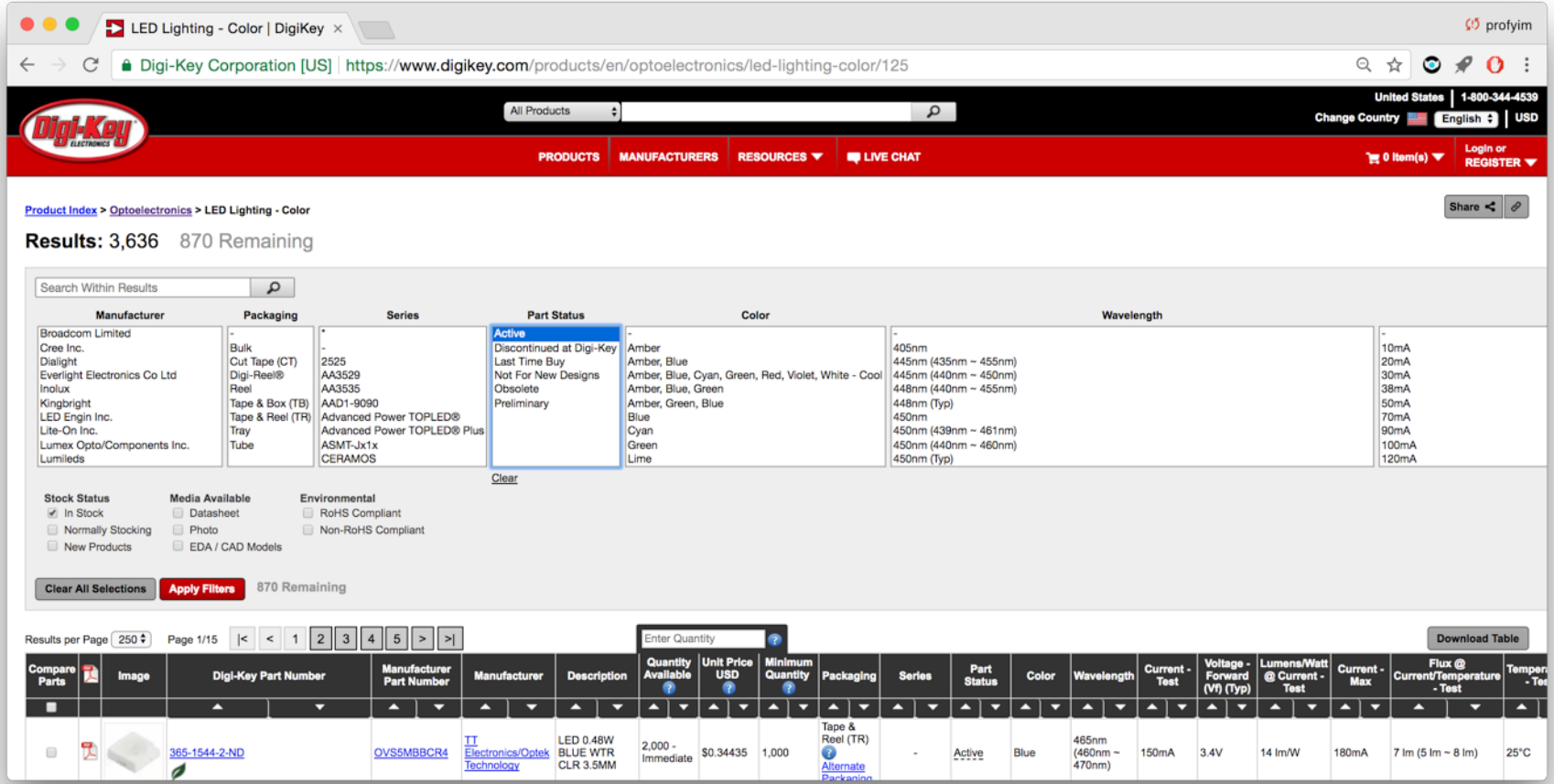
Aside – on coding 2;

From /usr/local/CrossPack-AVR/avr/include/**avr/sfr_defs.h** (on mac)

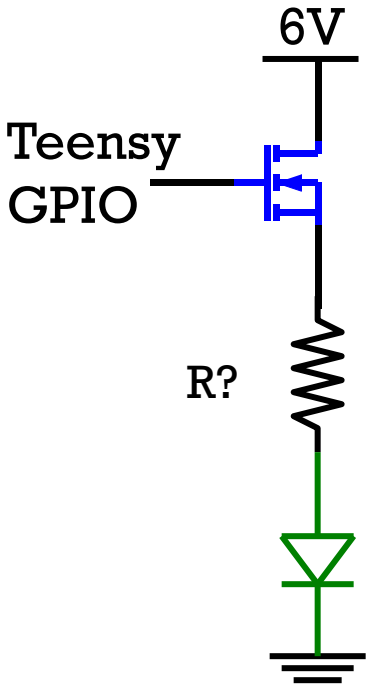
```
#define bit_is_set(sfr, bit) (_SFR_BYTE(sfr) & _BV(bit))
#define _BV(bit) (1 << (bit))
#define _SFR_BYTE(sfr) _MMIO_BYTE(_SFR_ADDR(sfr))
#define _MMIO_BYTE(mem_addr) (*(volatile uint8_t *)(mem_addr))
->
bit_is_set(sfr, bit)    ((*(volatile uint8_t *)(sfr)) & (1 << (bit)))
[mostly, skipping the part about casting the address pointer]

#define bit_is_set(sfr, bit) (_SFR_BYTE(sfr) & _BV(bit))
#define bit_is_clear(sfr, bit) (!(_SFR_BYTE(sfr) & _BV(bit)))
#define loop_until_bit_is_set(sfr, bit) do { } while (bit_is_clear(sfr, bit))
#define loop_until_bit_is_clear(sfr, bit) do { } while (bit_is_set(sfr, bit))
```

For C Casting tutorial see https://www.tutorialspoint.com/cprogramming/c_type_casting.htm

Digikey for LED



OSRAM GT PSLR31.13 (170 lumens)



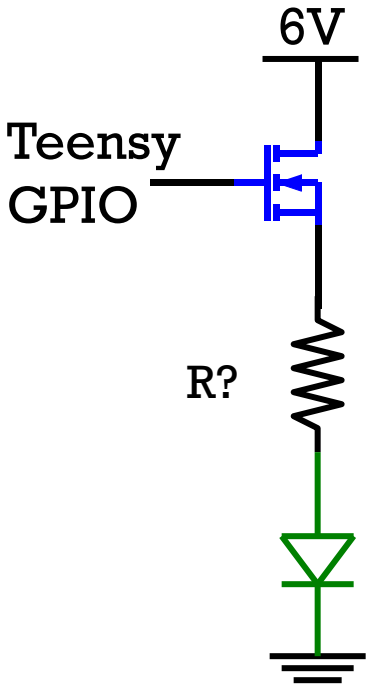
Q8: What value for R will maximize the brightness?

3mm x 3mm SMD

Parameter		Symbol	Values	Unit
Forward voltage 3) *page 21* Durchlassspannung 3) *Seite 21*	(min.) (typ.) (max.)	V_F V_F V_F	5.60 6.25 6.80	V V V

Parameter	Symbol	Values	Unit
Forward current Durchlassstrom (T_S = 25 °C)	I_F	10 ... 200	mA
Surge current Stoßstrom (t ≤ 10 µs; D = 0.005; T_S = 25 °C)	I_{FM}	300	mA

OSRAM GT PSLR31.13

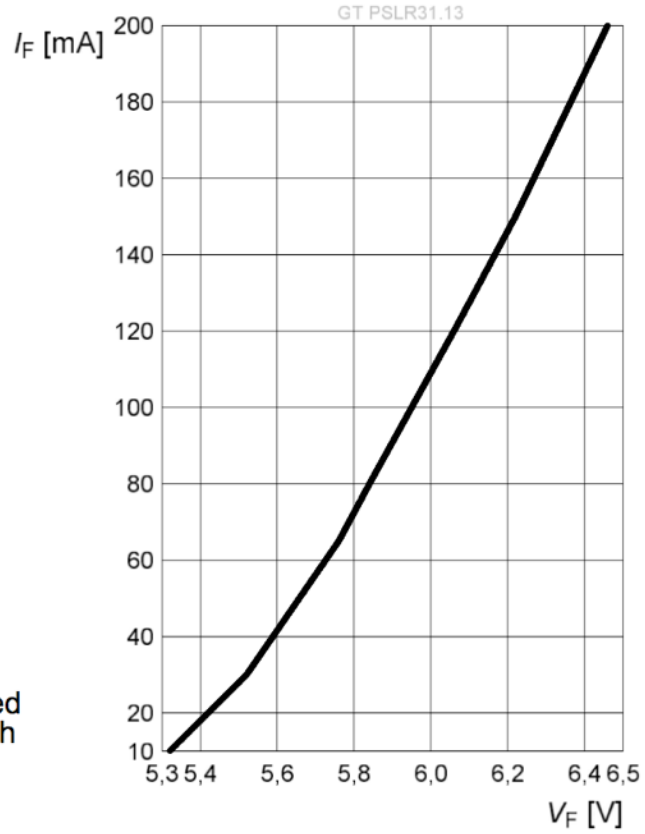


Forward Voltage Groups 3) *page 21*
Durchlassspannungsgruppen 3) *Seite 21*

Group Gruppe	(min.) V_F [V]	(max.) V_F [V]
B	5.60	5.80
C	5.80	6.00
D	6.00	6.20
E	6.20	6.40
F	6.40	6.60
G	6.60	6.80

3) **Forward Voltage:** The Forward voltage is measured during a current pulse duration of typically 1 ms with a tolerance of ± 0.05V .

Forward Current 5) *page 21* , 7) *page 21*
Durchlassstrom 5) *Seite 21* , 7) *Seite 21*
$I_F = f(V_F)$; $T_S = 25$ °C



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 SPI and I2C works

B. Using scanning for ToF or ultrasonic

C. Debugging sequence and trust