Lecture 21

Final Project/Game Rules
Autonomous Behaviors

Agenda

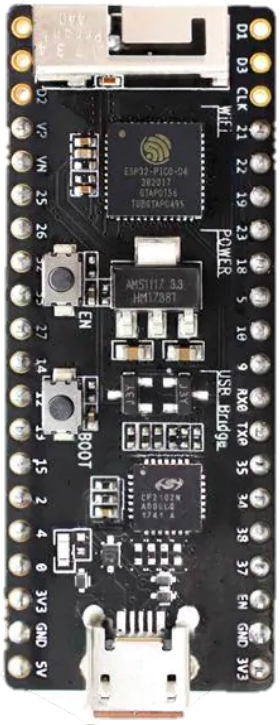
01. Game Rules
02. Behaviors (beacon tracking/wall following)
03. Lab4 Demo (Javascript code review) if we have time.

Stuff

- Submitting expense report for reimbursement. Some updates to webpage – wait a day for system to update. New *Expense Reports-SEAS Guide.pdf* now on canvas resources to help.
- Late days can be applied to Final project graded Evaluation. For team projects 1 late day equiv to 1/3 late day of individual.
- ESP32's are running low. Worldwide supply chain issues hitting us. We do have some stock.
- T-shirt size form will go out today. Need to submit by end of week

ESP32 Board Variants we have (some)

~$10
No controllable LED



PicoKit

~$10
Pin labels underneath

HiLetGo
NodeMCU

Newer, better board $8
single core, no FPU, no BT
Too fat for breadboard

We have 100 of these in stock.

Be sure to set different board in Arduino.

I'm not 100% everything works with this board

Better Schottky diode
Better LDO regulator

ESP32 S2
Saola 1R

Final Project Tentative Schedule *Option 1*

14	15 Today	16	17	18	19	20
21	22	23	24 Design Review 1	25 Thanksgiving	26 break	27
28	29	30	1	2	3 Design Review 2	4
5	6	7	8	9 Optional Graded Evaluation	10 Last day of classes	11 Reading days
12 Reading days	13 Reading days	14 Reading days	15 Graded Evaluation	16 Graded Evaluation	17 Graded Evaluation	18 Group Stage
19 Final Rounds	20	21	22 Last day to submit	23	24	

Final Project Tentative Schedule *Option 2*

14	15 Today	16	17	18	19	20
21	22	23	24 Design Review 1	25 Thanksgiving	26 break	27
28	29	30	1	2	3 Design Review 2	4
5	6	7	8 Graded Evaluation	9 Graded Evaluation	10 Last day of classes	11 Game Reading days
12 Game Reading days	13 Reading days	14 Reading days	15	16	17	18
19	20	21	22 Last day to submit	23	24	

Final Project Tentative Schedule

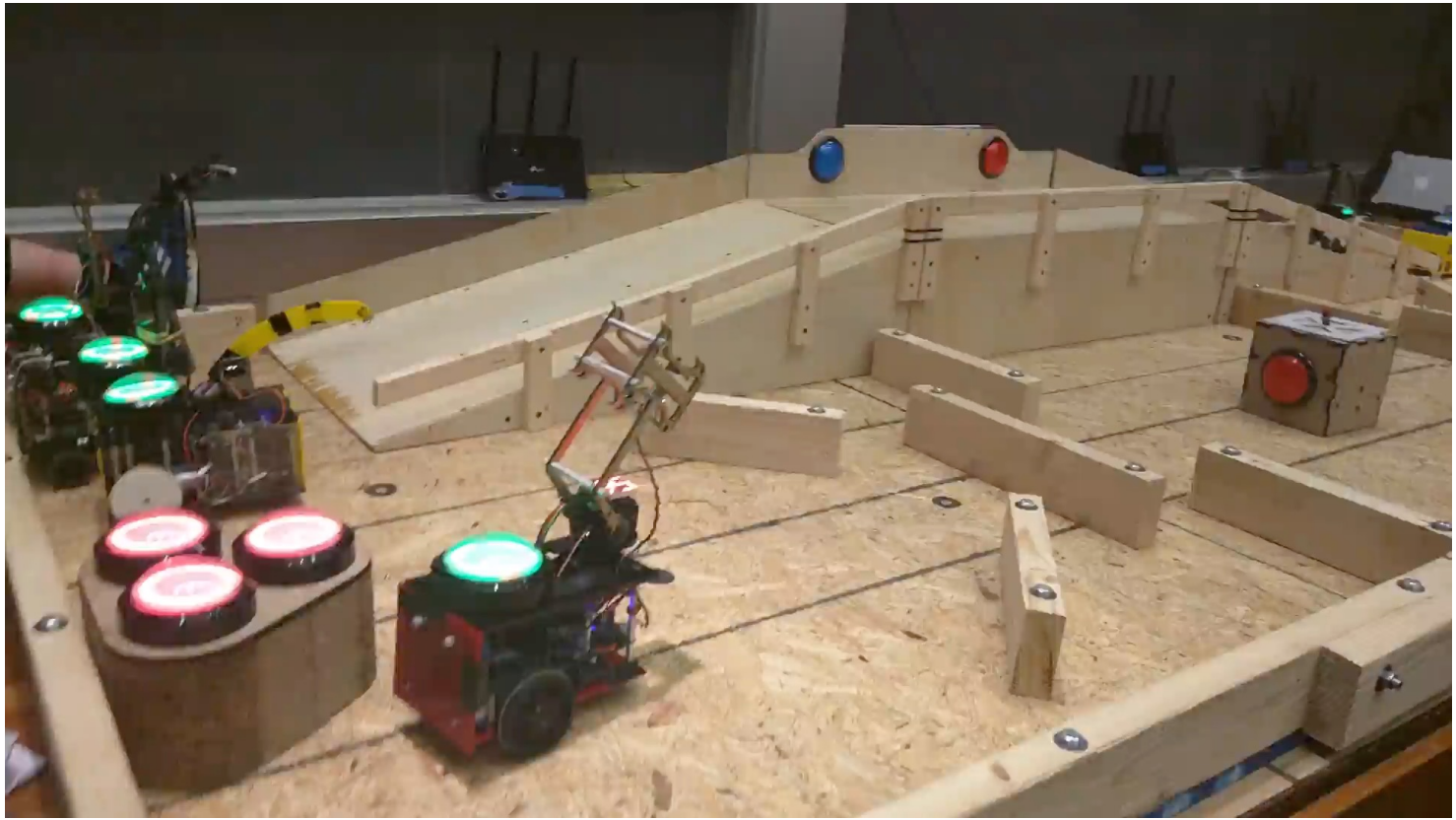
14	15 Today	16	17	18	19	20
21	22	23	24 Design Review 1	25 Thanksgiving	26 break	27

- For this design review that is just before Thanksgiving, how many people prefer to do the design review by zoom? (Raise your hand in class, vote yes or no in chat)
- How many people prefer to do both design reviews by zoom? (Raise your hand in class, vote yes or no in chat)

01

Game Rules

MEAM510 - **2018** Game



Grand Theft Autonomous 2021C

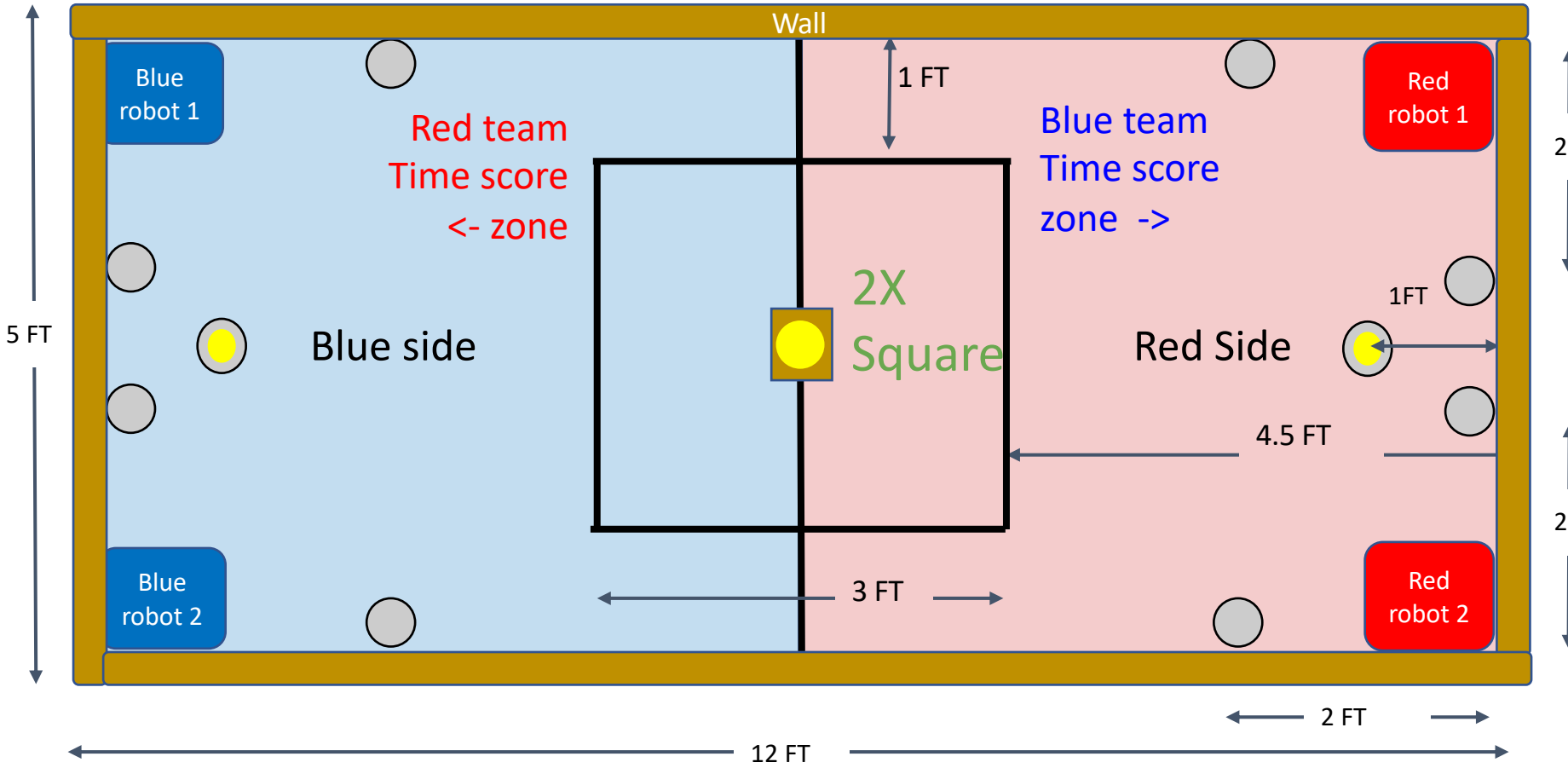
- The goal of this game is to get as many points as you can. Ideally by stealing soda cans from your opponent.

- Game is played head-to-head with teams of two robots (e.g. 2v2)

- Teams move robots semi-autonomously to move or steal cans from each other to score points. The team with the most points when time expires wins.

Field of play

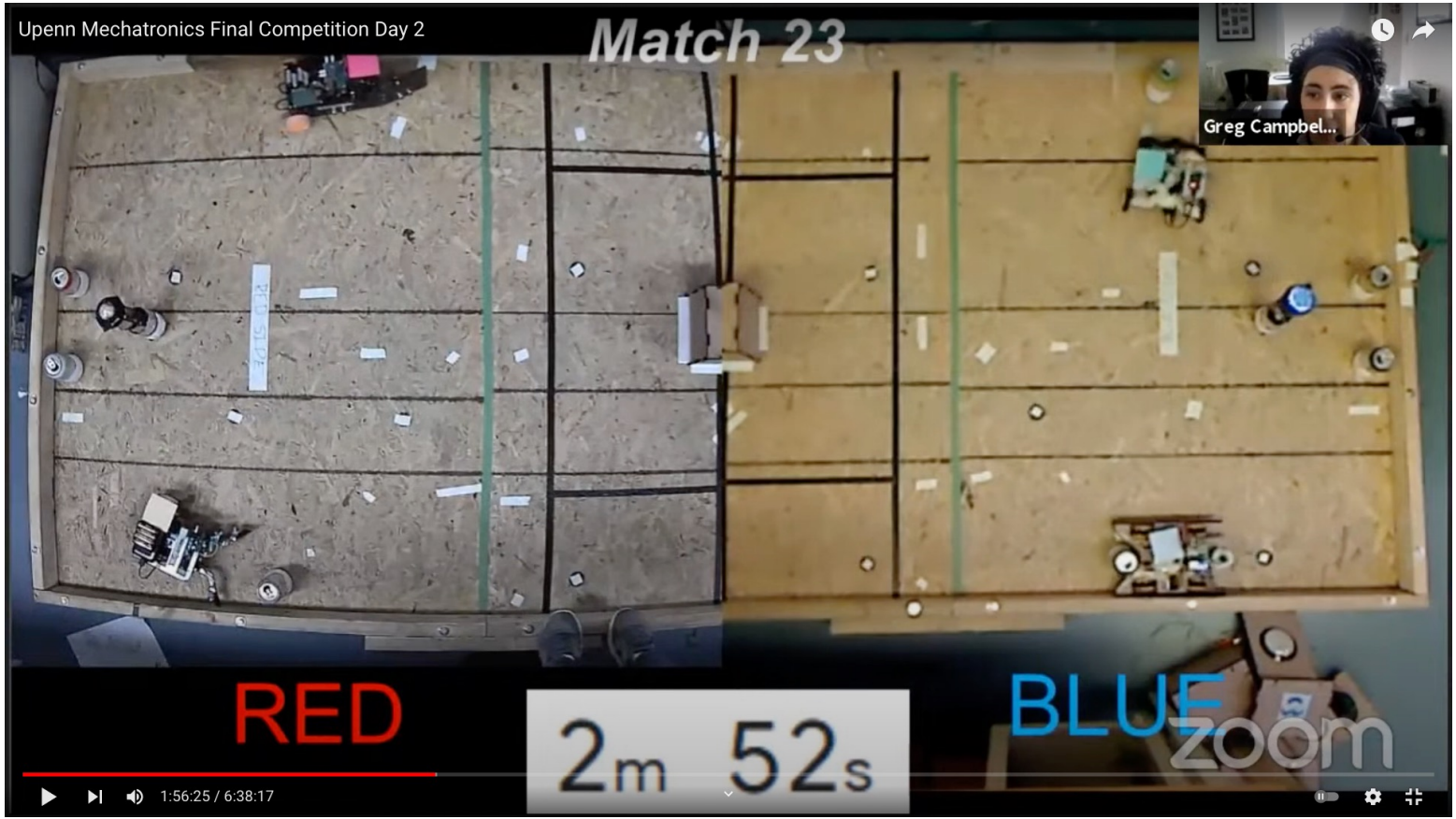
Approximate dimensions +/- 5%
Drawing not to scale
Field is symmetric vertically and horizontally



Game Play

- Each side has their own zoom with view of their own half. Teams can communicate with each other and TA's but not other side.

MEAM510 – **2021a** Game



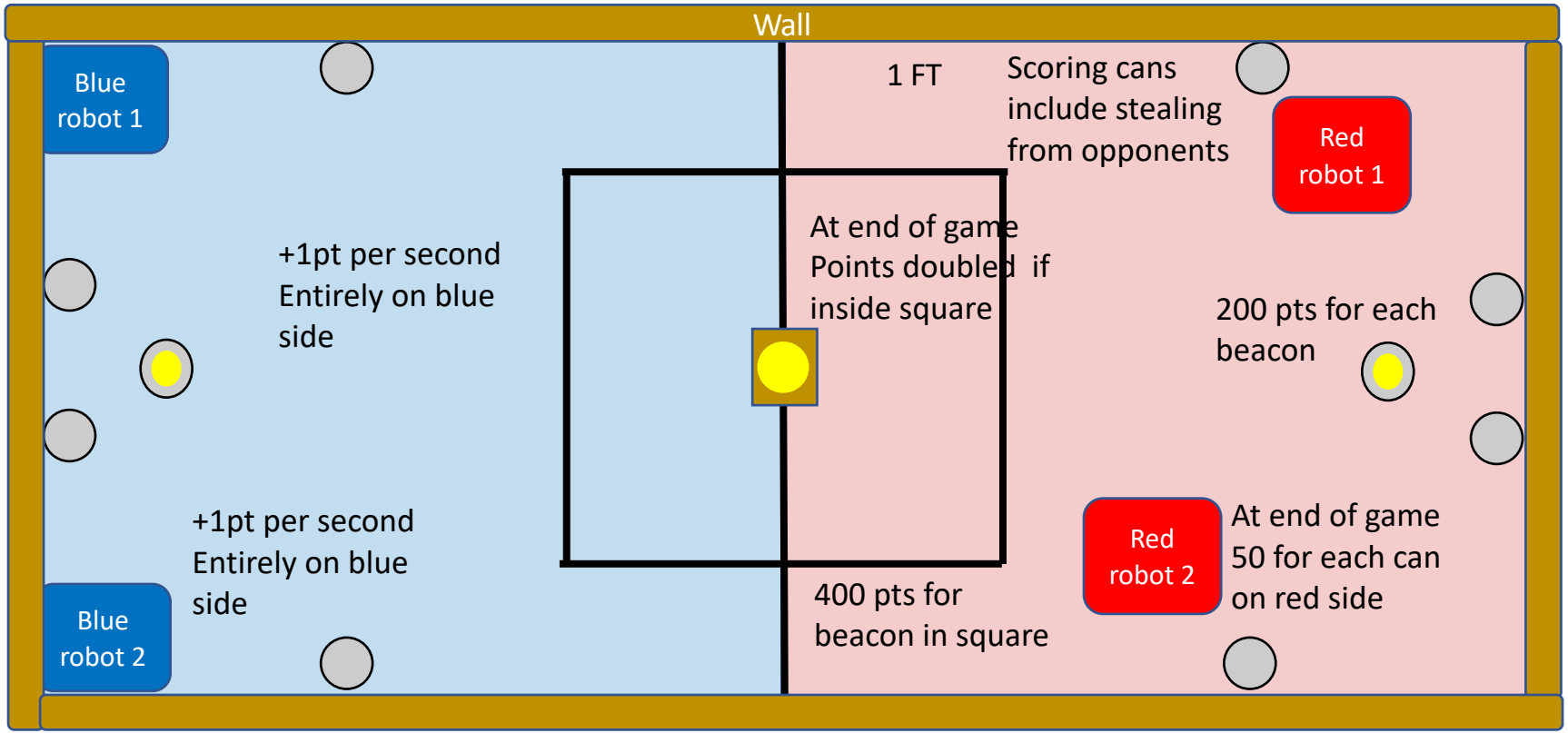
https://sites.google.com/seas.upenn.edu/meam510-2021a/highlights

Difference from 2021A game

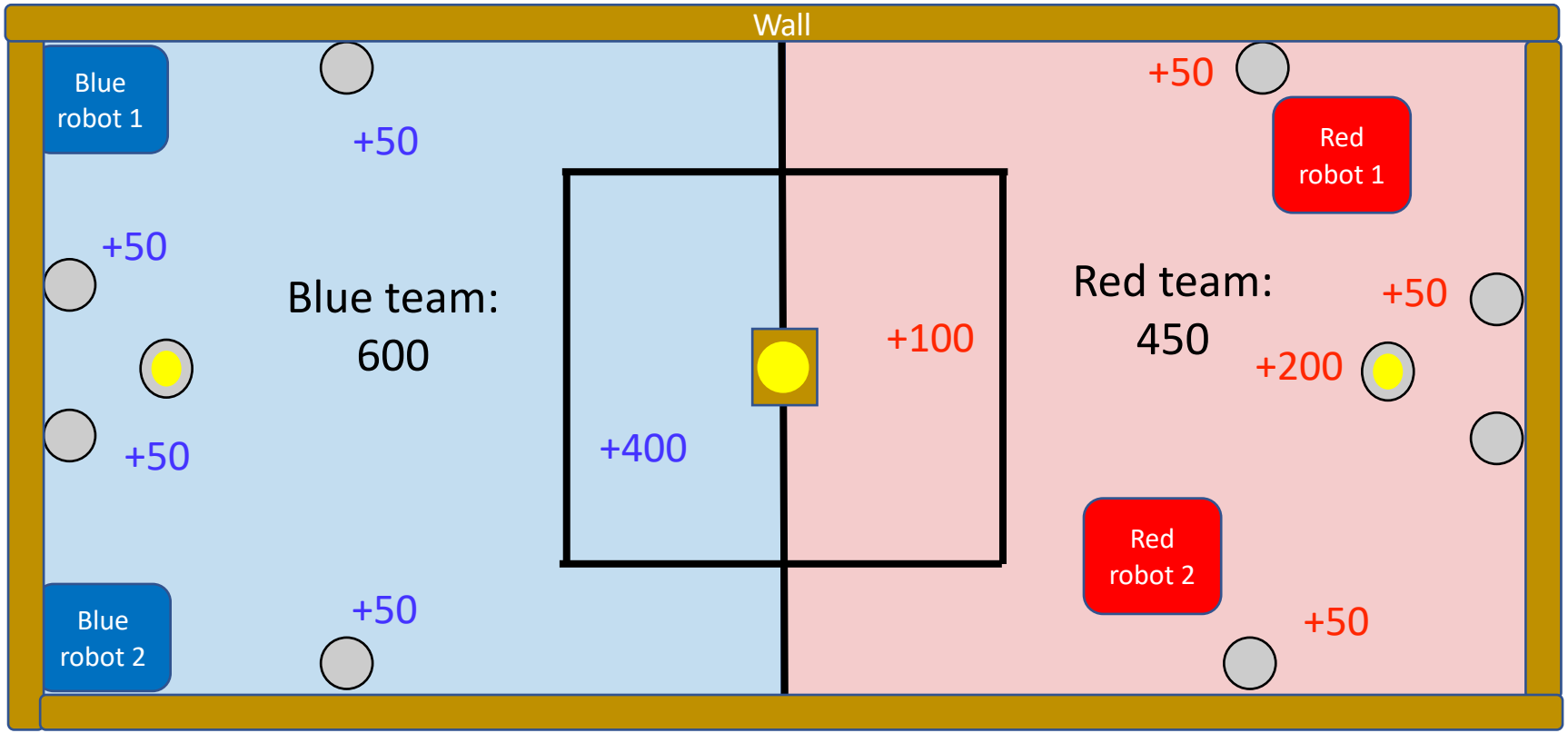
- HTC Vive localization. Objects and robots can determine their own positioning relative the the Vive lighthouse
- Robots must transmit their own position via UDP
- Time-based scoring depends on **both** position on enemy side and no wireless communication.
- Logging/monitoring communication through wireshark
- Position of objects is transmitted to everyone via UDP

Blue team view Overhead Cam Red team view

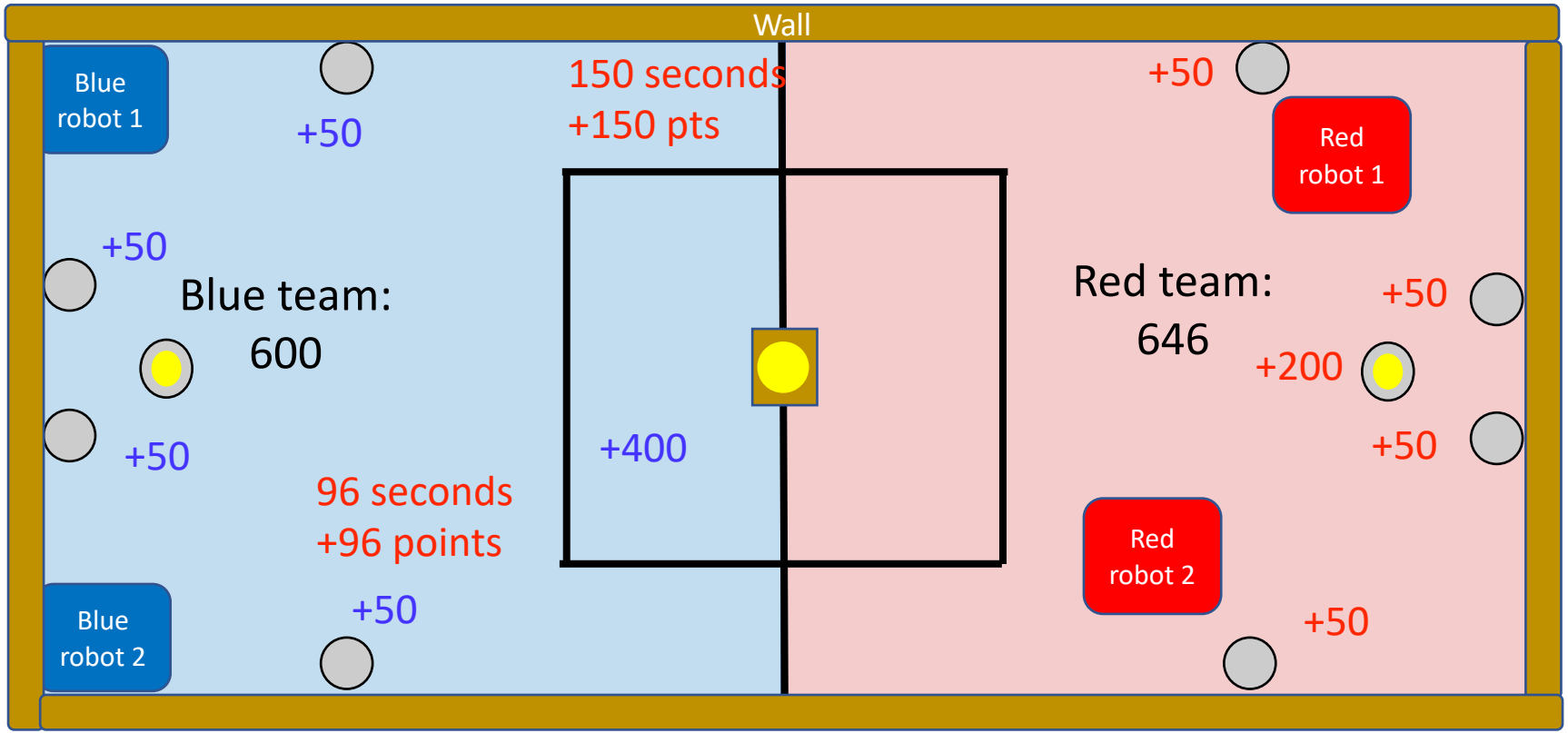
Scoring: Timing points and object points



Score example: no blind excursion, no timing pts



Score example: red team autonomous time points



Scoring summary

- Object points starting position: 400pts 400pts
- Object points max none stolen: 800pts 800pts
- Object points max all stolen: 1600pts 0 pts

- Max time points for 3 minutes two robots: 360 pts

- Min total score for both teams combined 800 pts
- Max possible score for one team 1960 pts

Q1: What is the maximum total score for both teams combined?

Robot Constraints

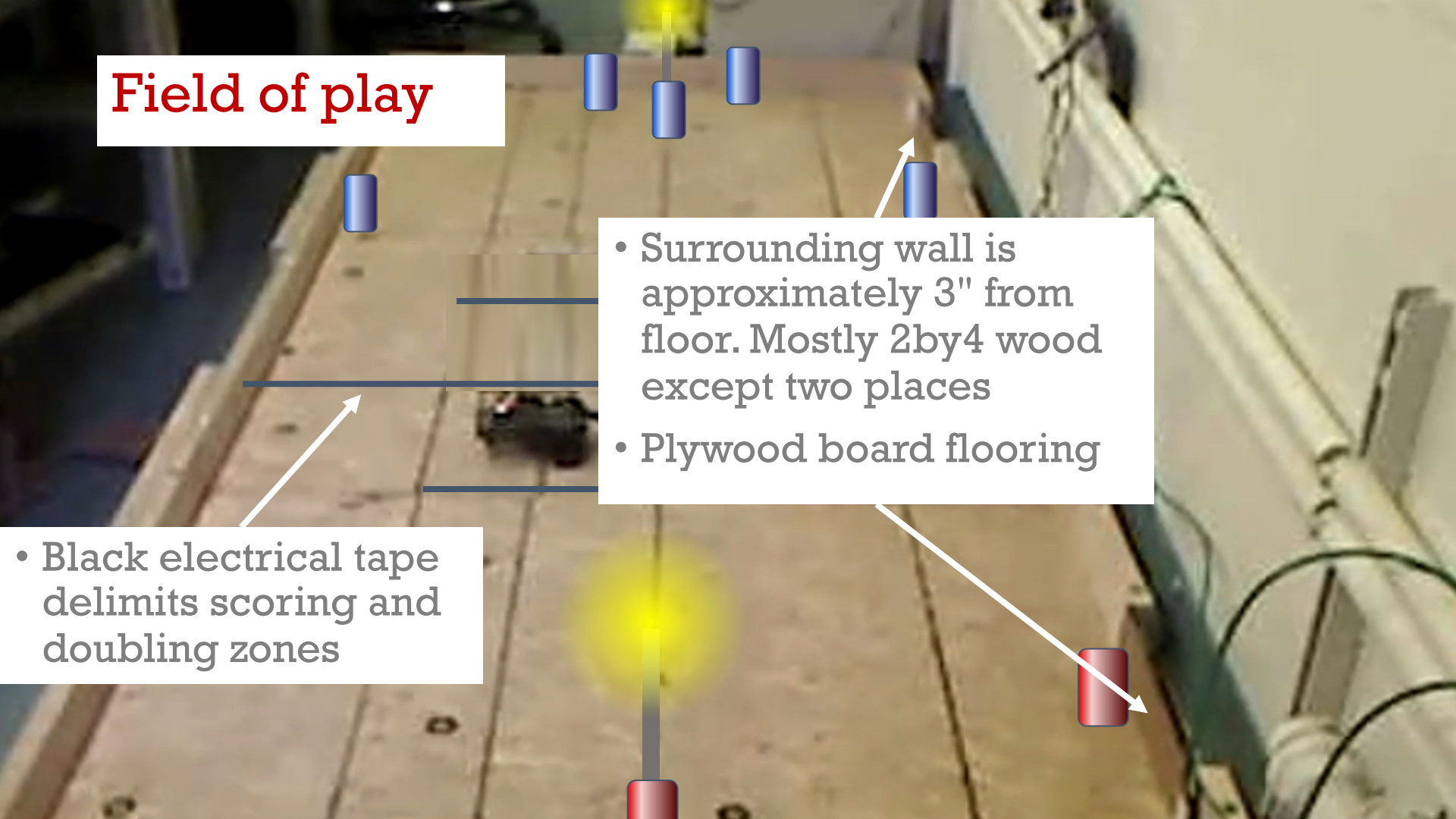
- Robots must fit in 12” x 12” x 12” box to start
- Robots are controlled via internet through one URL via ngrok
- Robots can have a maximum of 10Kbytes/sec data transmission each way
- Robots can’t physically damage other robots or field
- Robots may not intentionally disrupt sensing or communication of other robots.

Robot Grading: Functional evaluation

Robots will be placed on field alone (no other robots) and tested for the following: Must be achieved before game time expires.

- For receiving passing grade in this class
  - User control of mobile base to directed targets both with direct vision of robot and without.
- For full marks on Final Project
  - Identify and move towards (either 23Hz or 700Hz beacon)
  - Perform wall-following autonomously.
  - Use the lighthouse to transmit your X-Y location via broadcast UDP
  - Move to a TA-given X-Y location automatically
  - Blindly move to a can transmitting its XY location

Field of play



- Surrounding wall is approximately 3" from floor. Mostly 2by4 wood except two places
- Plywood board flooring

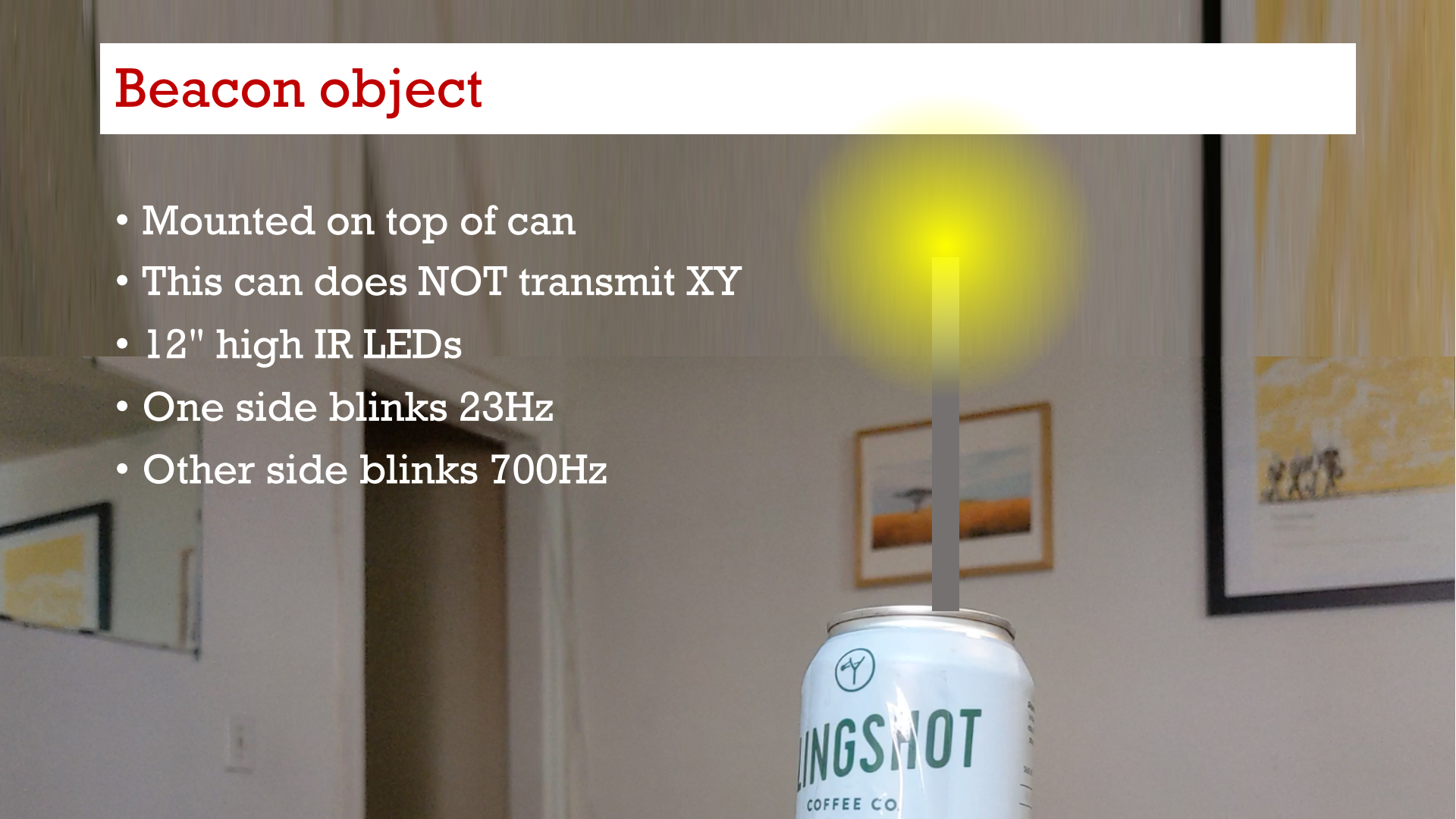
- Black electrical tape delimits scoring and doubling zones

Soda can objects

- Weighted to stay upright (mostly)
- X, Y location of each can is transmitted once per second via UDP broadcast

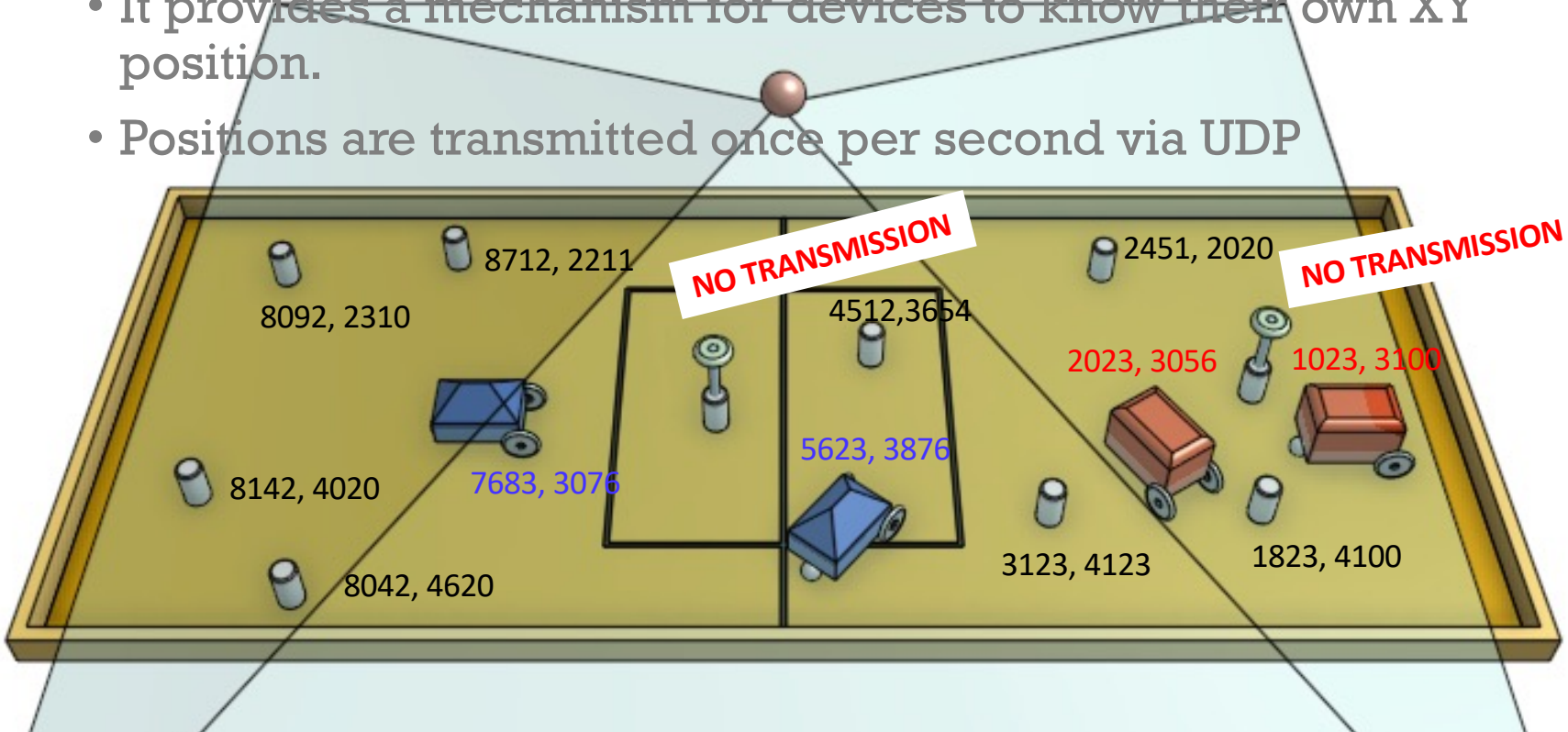
Beacon object

- Mounted on top of can
- This can does NOT transmit XY
- 12" high IR LEDs
- One side blinks 23Hz
- Other side blinks 700Hz



Vive Localization

- We will have a full lecture on the Vive on Wednesday
- It provides a mechanism for devices to know their own XY position.
- Positions are transmitted once per second via UDP



Frequently Asked Unusual Questions

- Can we make something that grabs an opponent?
  - Yes! Something like Blitzcrank would be awesome - but hard…
- Can we make a quadrotor drone?
  - Yes… if you can make one in our budget and control it with 10k byte/sec.
- Can we make a jumping robot?
  - See above
- Can we throw things at opponents or cans or throw cans?
  - Yes as long as it is not dangerous
- Can we make a giant arm?
  - Yes if it starts in the 12"x12"x12" constraint.

Q&A on Game Grading or Game Rules?

Purchase recommendations

- Choosing motors (pololu, sparkfun) maybe largest purchase.
- If high current motor is used, may want motor driver - can make things easier than building your own h-bridge.
- Wheels and motor shaft mounts may be a good purchase. Extra batteries are an option. Don't rely on AA's supplied by GMlab (purchase your own w/project funds)

- Delivery time:
  - McMaster typically 3 business days from order.
  - Digikey typically 3-5 business days from order.
  - All others 5-10 business days from order.
- Don't buy a "robot kit". This is against the rules.

Recommendations

- Robot – robot contact is likely. Strong heavier robots tend to do better in a pushing contest.
- Stealing objects from the other side causes large score swings.
- Doubling square can lead to easy steals.
- Defense can be important but will not win a game (no points gained).
- Don't try to do too much.
- Develop strategy for doing minimal first adding more functionality as time allows.
- Place orders EARLY (expect shipping delays)

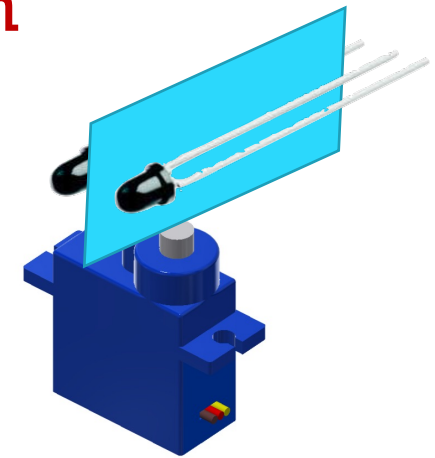
Potentially Useful Behaviors/sensing?

Q2: List 4 different potentially useful behaviors and associated sensing modalities

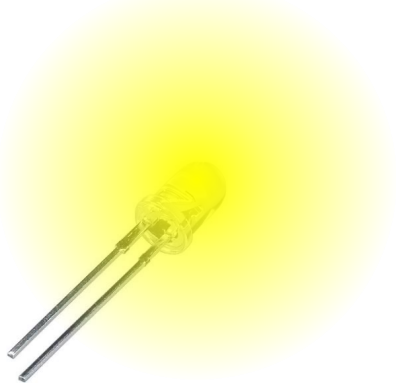
02 Behaviors

Beacon Tracking

Beacon Tracking Architecture Option



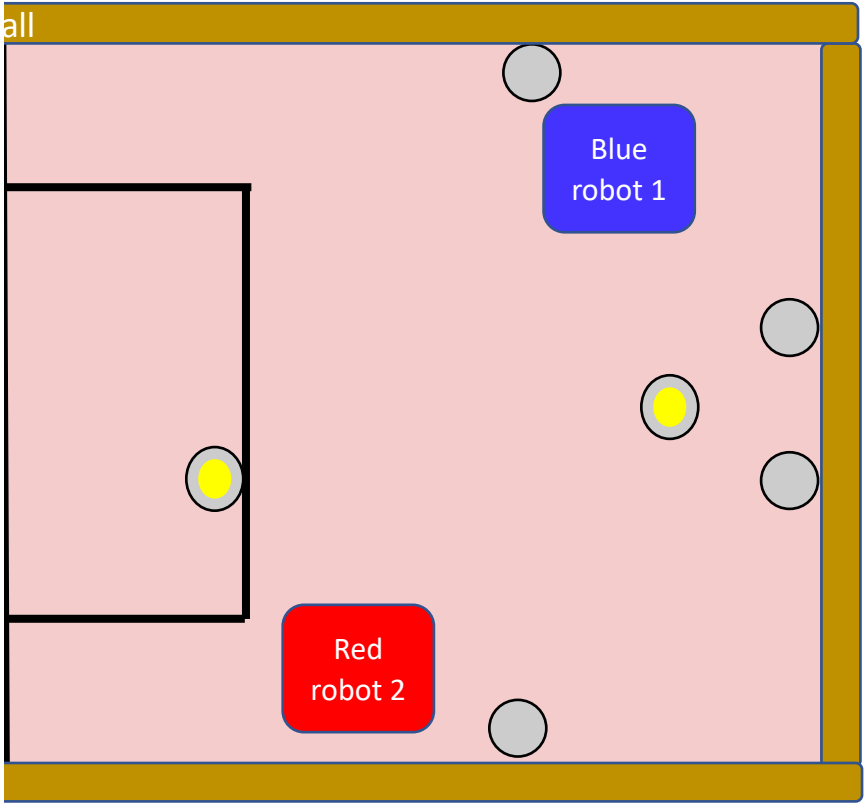
- Mount two IR phototransistors on a servo with optical separator
- If the left side sees the beacon rotate left
- If the right side sees, then rotate right
- Keep track of servo position to steer robot.



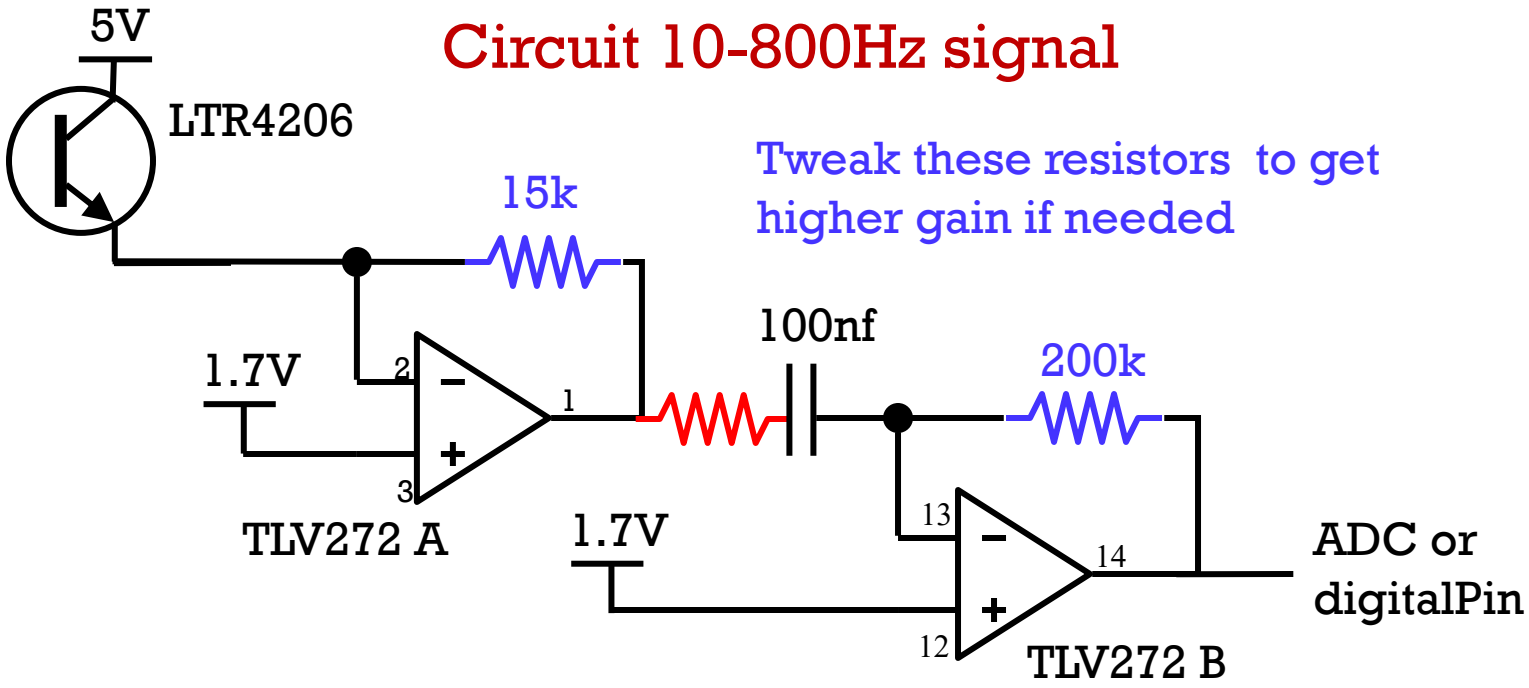
Example behaviors

- Lock on beacon,
- Align robot with servo.
- Move forward until contact

- Lock on beacon
- Align robot 90 to servo,
- Move forward while turning to maintain 90 servo angle



Circuit 10-800Hz signal

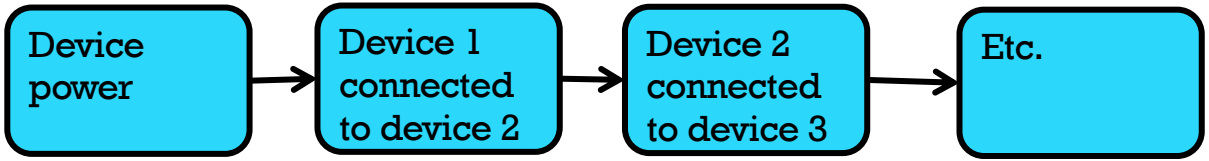


Q3: Whats the difference between doubling the 15k to 30k vs doubling 200k to 400k?

Q4: When and why would you use digital pin instead of ADC?

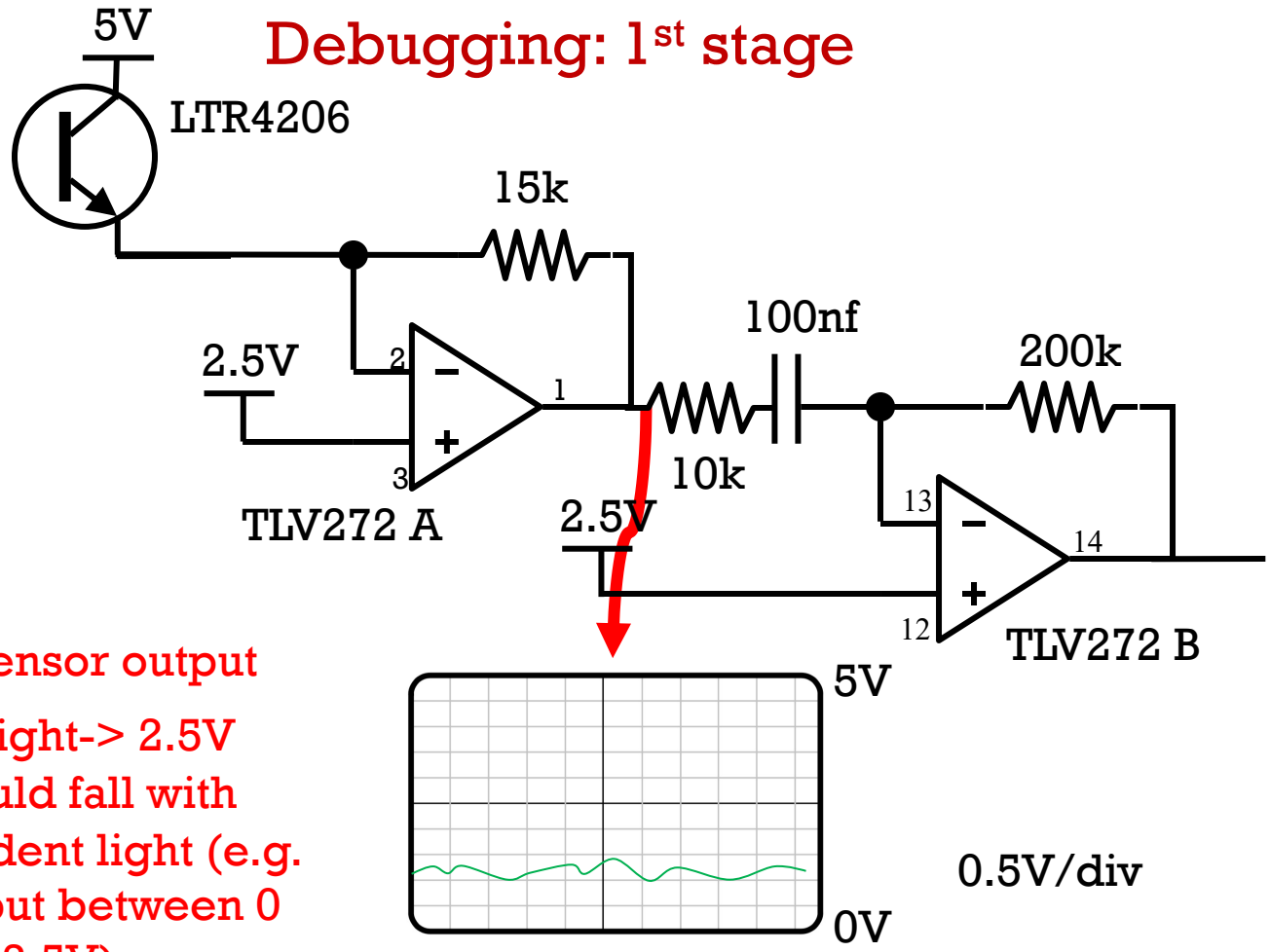
Debugging Electronics

- Hardware and Software piece - need to check both
- Break your system apart into pieces
  - viewed as linked via inputs and outputs
  - Ideally a chain of parts with 1 output from one part going to 1 input in the next part.



- Check known voltage at each part starting with power and ground.

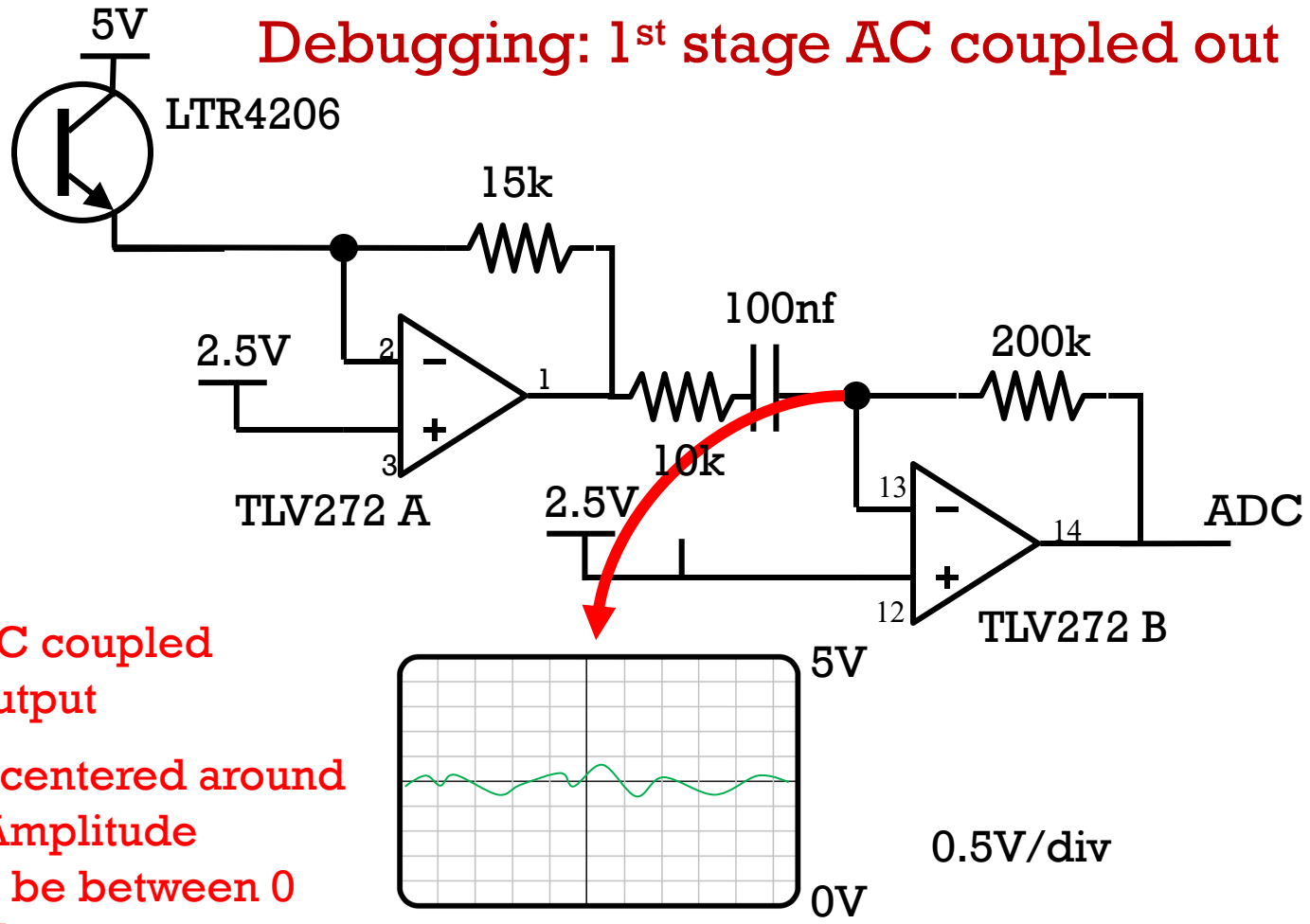
Debugging: 1st stage



Sensor output

No light-> 2.5V
Should fall with incident light (e.g. output between 0 and 2.5V)

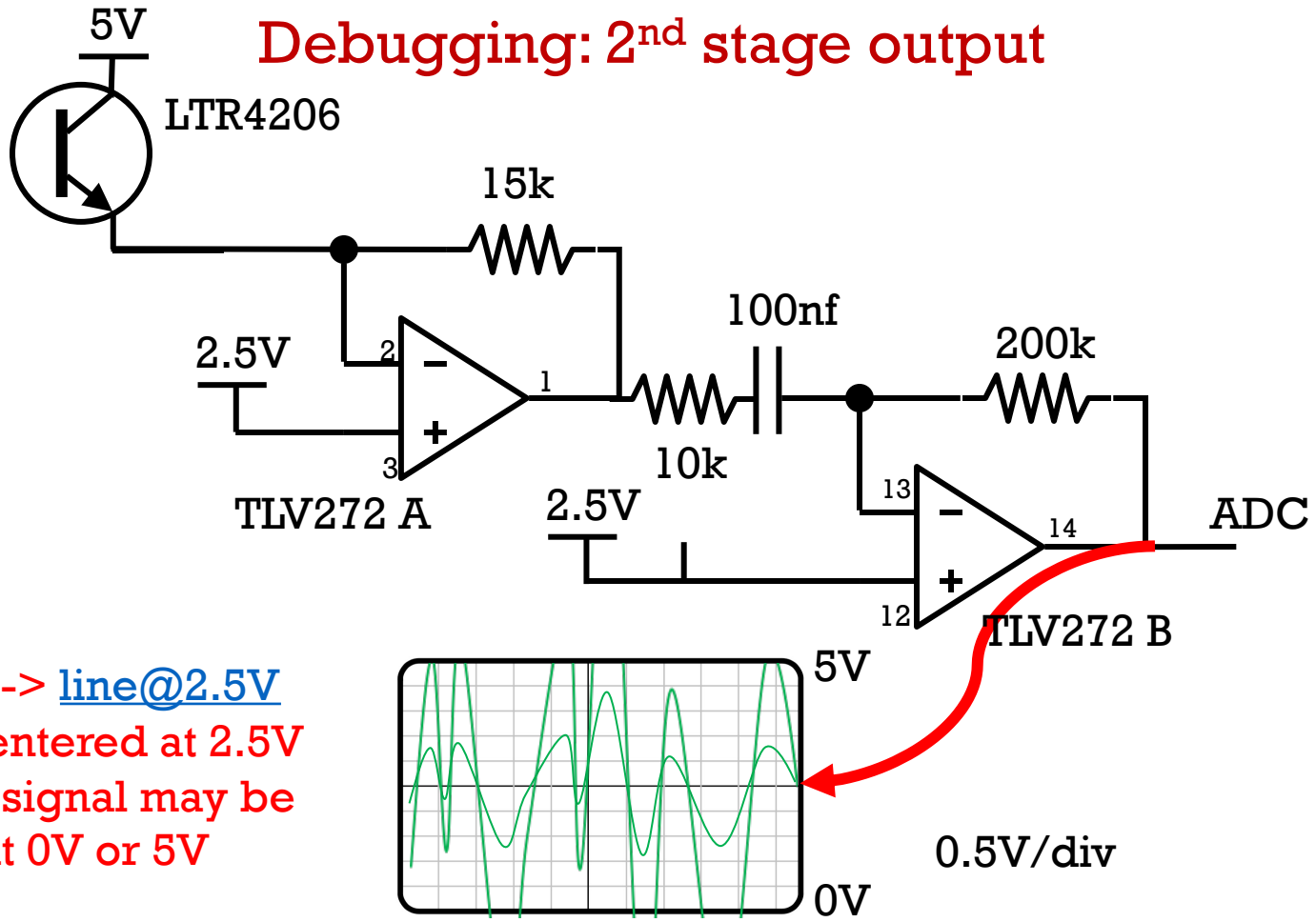
Debugging: 1st stage AC coupled out



AC coupled output

Signal centered around 2.5V. Amplitude should be between 0 and 5V

Debugging: 2nd stage output



No signal-> line@2.5V
Ideally centered at 2.5V
But large signal may be clipped at 0V or 5V

Bang-bang Beacon Track Code example

```
//You need setup() and other
//You need #defines PERIOD, NOISE, RES, SERVOOFF etc.

uint32_t us;
int x;
int pin[] = {PIN1, PIN2};

void checkFreq(int ch) {
  static int oldpin[2];
  static uint32_t oldtime[2];

  if (mydigitalRead(pin[ch]) != oldpin[ch]) {
    int per = us-oldtime[ch];
    if (per>PERIOD-NOISE && per<PERIOD+NOISE) {
      if (ch==0)   x++;
      else x--;
    }
    oldpin[ch] = mydigitalRead(pin[ch]);
    oldtime[ch] = us;
  }
}
```

```
void loop()
{
  static uint32_t lastServo = micros();

  us = micros();
  checkFreq(0);
  checkFreq(1);

// update the servo position
  if (us-lastServo > 1000000/10) {
    Serial.printf("%d servo x=%d \n", x);
    ledcAnalogWrite(0, SERVOOFF+x, RES);
    lastServo = us;
  }

  delayMicroseconds(200);
}
```

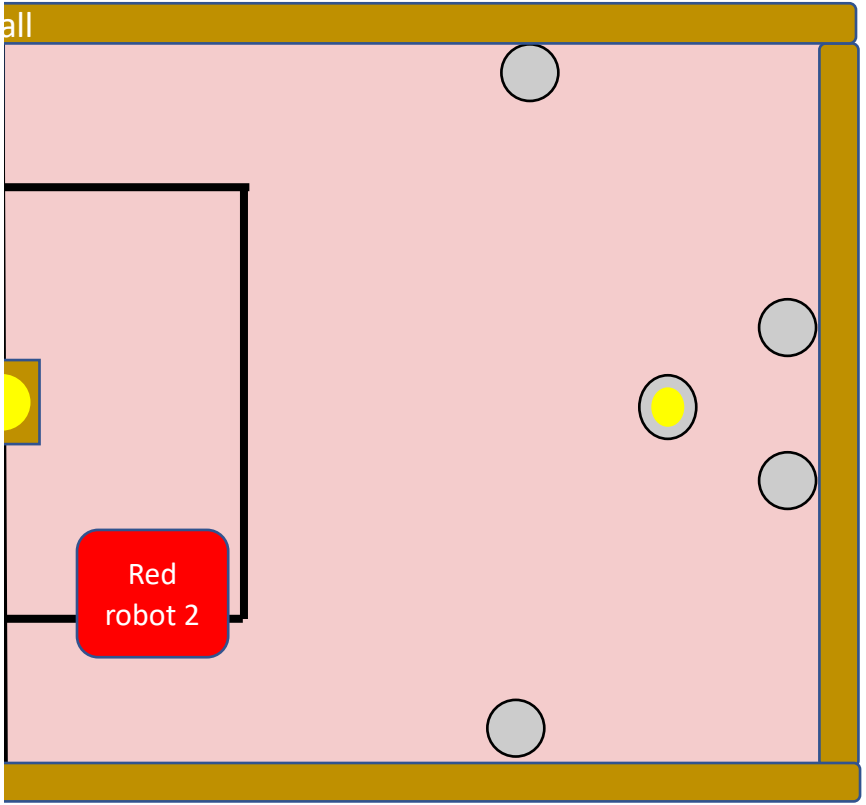
02b

Wall Following Behavior

Wall Following

- Behavior could be very useful.

- Sense the wall (either through contact or through range sensing)
- Constant forward motion, maintain distance from wall, turning right towards wall, turning left away
- If something in front, then turn left (ignore cans and beacons?)
- What happens if another robot is there?



Wall sensing options (discussed next week)

- Contact sensing (whisker) through
  - switch
  - capacitive sensor modified into switch
- Distance sensing
  - IR proximity sensing
  - Ultrasonic ranging
  - Lateral effect photodiode
  - Time of Flight

02c

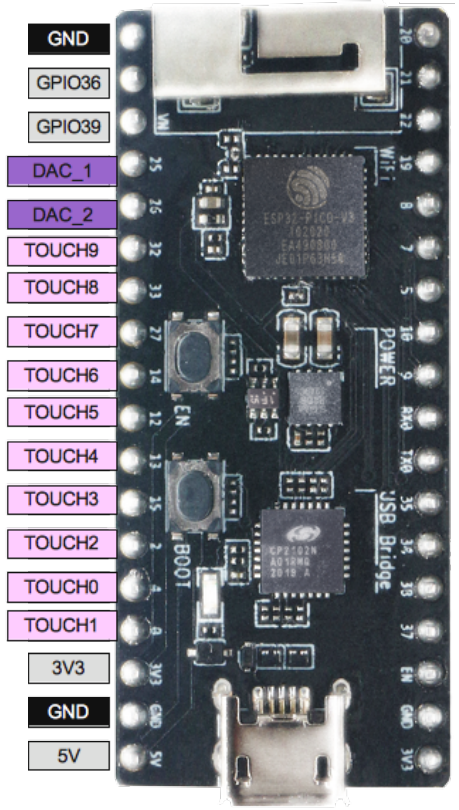
ESP32 Capacitive Sensor

Capacitive Touch Sensor

- One wire "circuit."
- Capacitance is measured by generating frequency on output pin then reading voltages with ADC.
- Use Arduino command

```
int touchRead(TouchPIN);
```

- Returns values around 50-100
- Values will drift, but sudden drops in value usually mean contact.



Maybe useful as conductive contact sensor

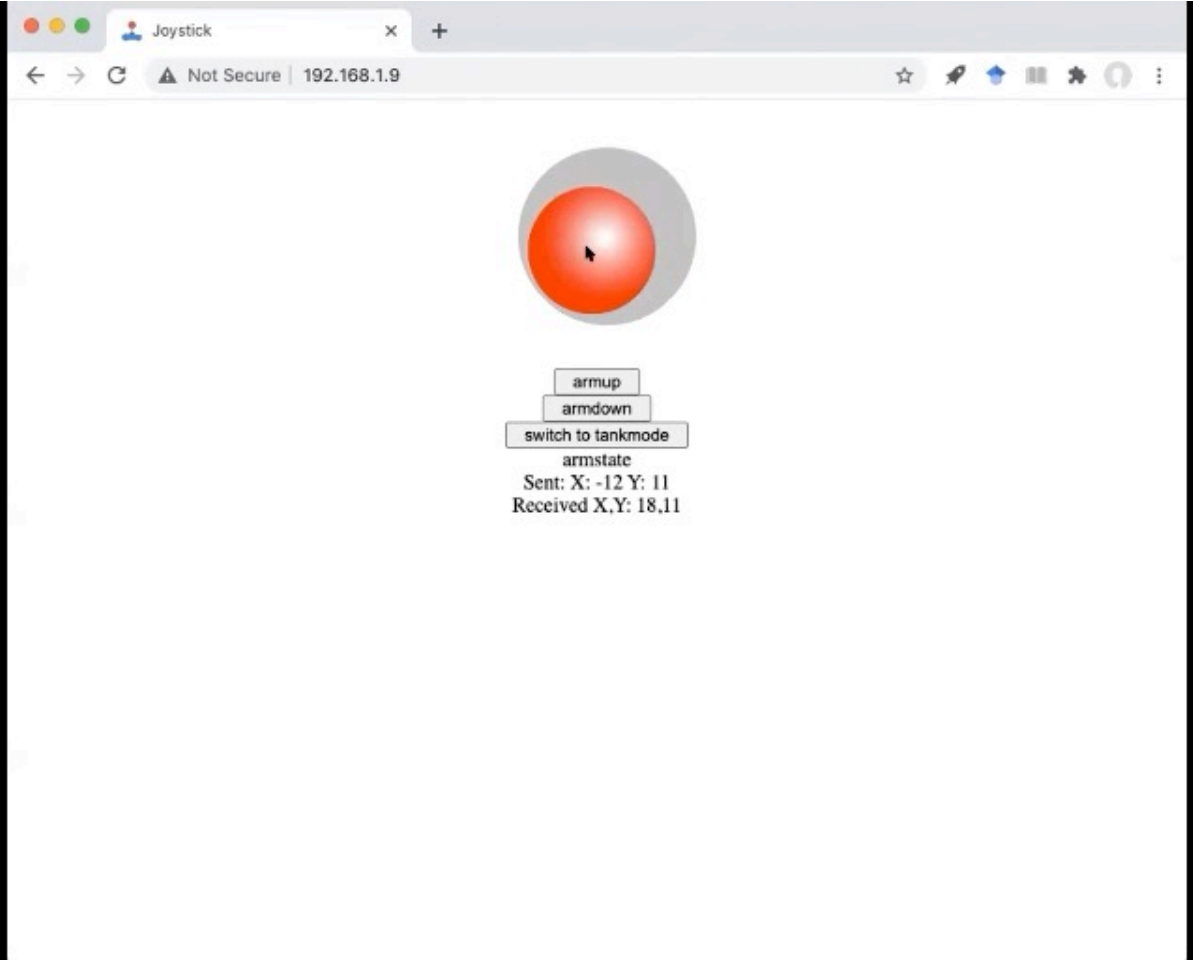
- Distinguishes between touching can vs anything else
- Feelers can be bare wires.
- For two-wire contact (larger response) need to have mechanism for both wires to contact object but not each other.
  - Ideally compliant wires so if one makes contact it doesn't stop the other from making contact
  - But not so compliant that the two wires touch each other.

Summary

- Two low level autonomous behaviors are required for the final project: wall-following and beacon tracking
- Some tweaking required to get it to work reliably both for control and sensing.
- Capacitive touch sensor is a way to detect cans
- Op amps are better way to detect small signals than phototransistors and resistors alone.

03

Lab4 Demo Code (from last year)


Joystick
Not Secure
192.168.1.9
armup
armdown
switch to tankmode
armstate
Sent: X: -12 Y: 11
Received X,Y: 18,11

TankMode

Not Secure | 0a874c2fe144.ngrok.io

Tank mode: Use Keys [Q] [A] and [P] [L] for levers.
[S] for left arm and [K] for right arm

[S] left button right button [K]

Command Sent: lever?val=0,0,0,0

ESP32 Received: 0,0,0,0

switch to Joystick

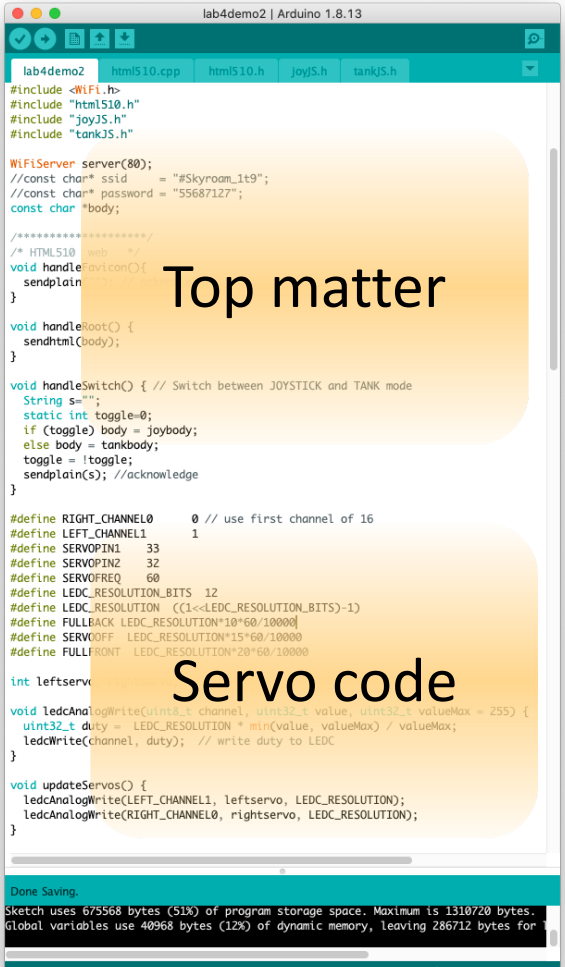
a

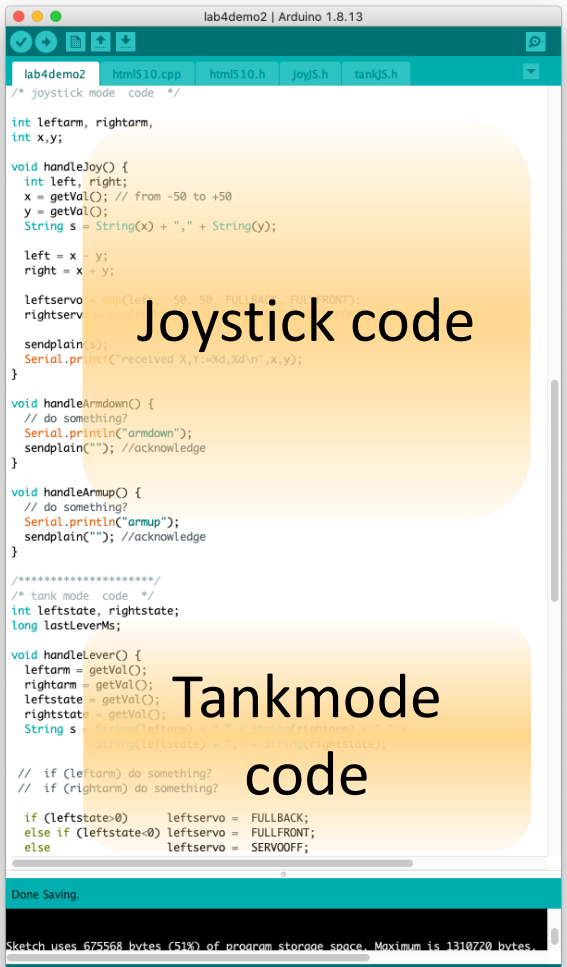
Lab 4 DemoCode structure

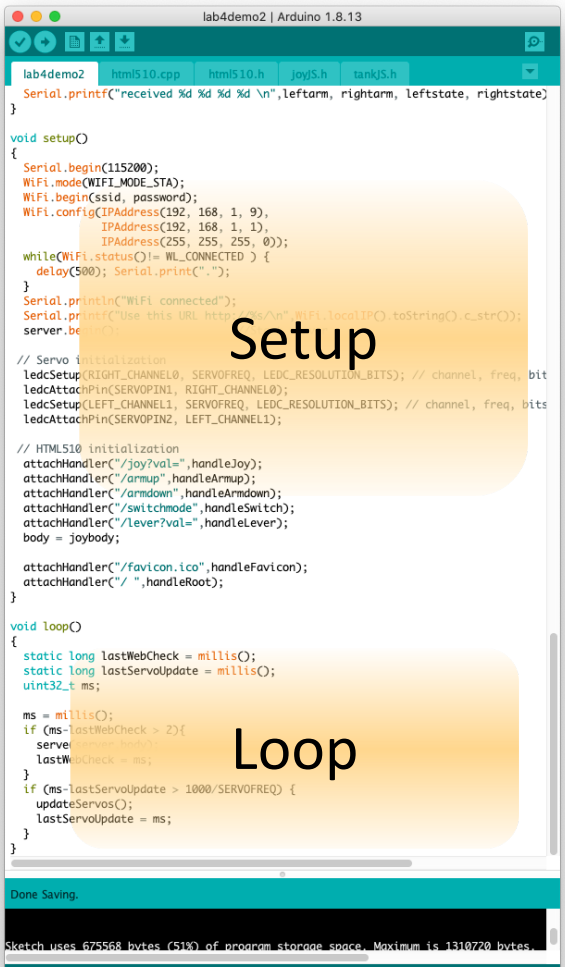
lab4demo2.ino	• High level Arduino Modify this code for your robot
tankJS.h	• Javascript code for tankmode interface
joyJS.h	• Javascript code for joystick mode
html510.cpp	• Old version linking ESP32 and HTML
html510.h	• Old version header support for HMTL

Code on Canvas uses older incompatible version of html510.cpp

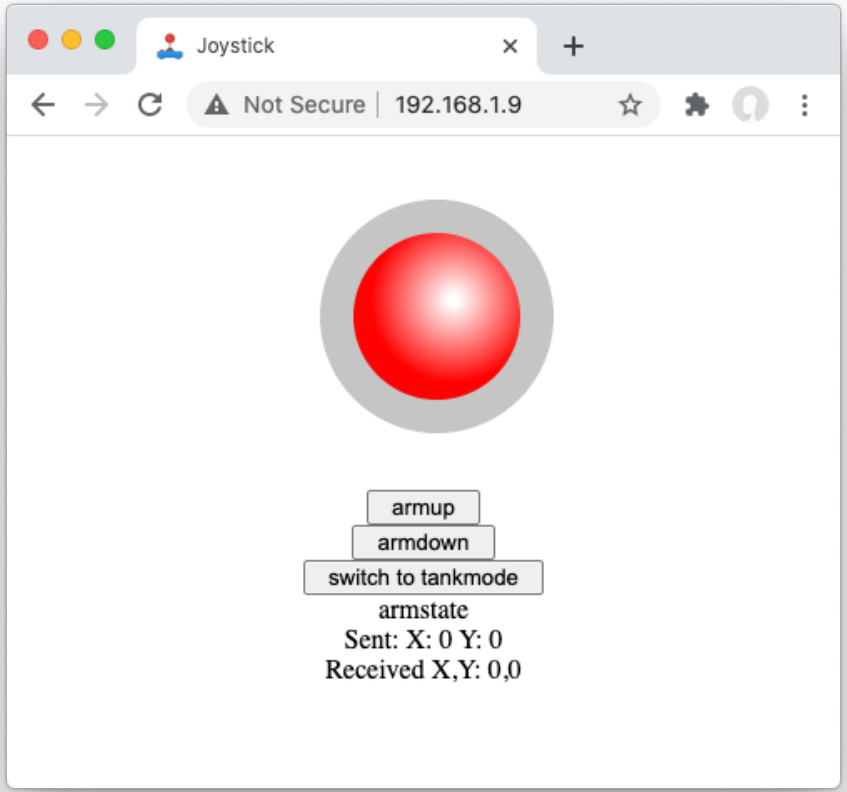
lab4demo2.ino



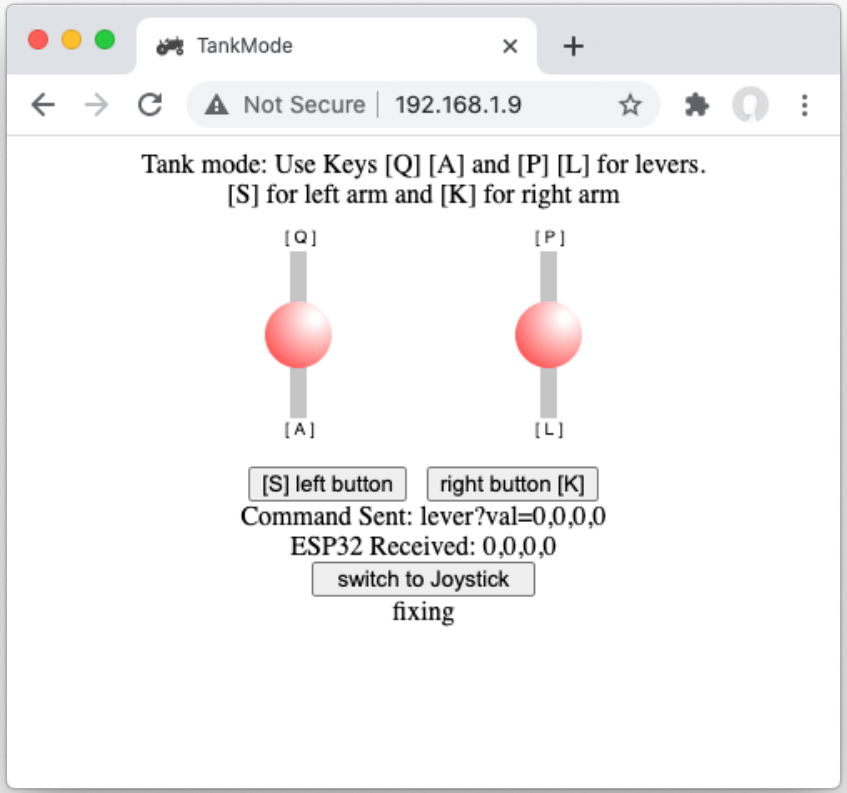




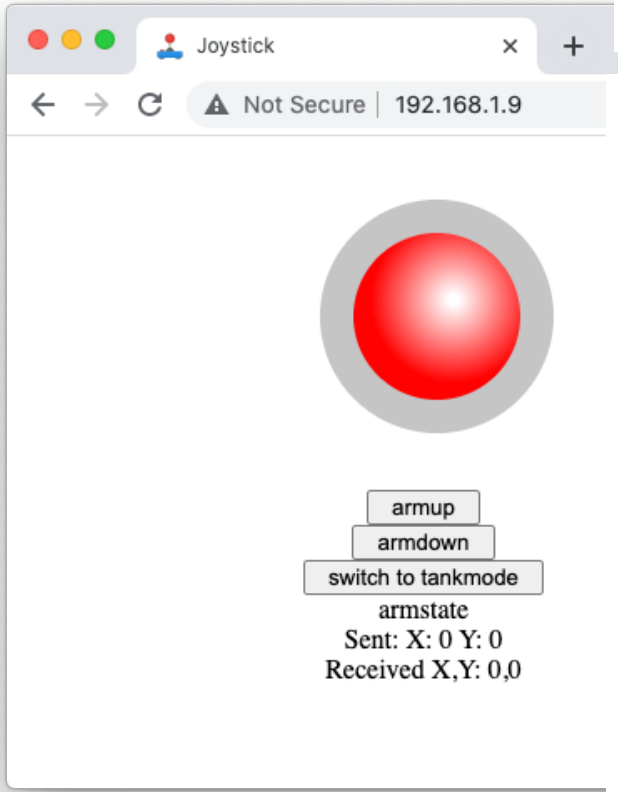
joyJS.h



tankJS.h



joyJS.h



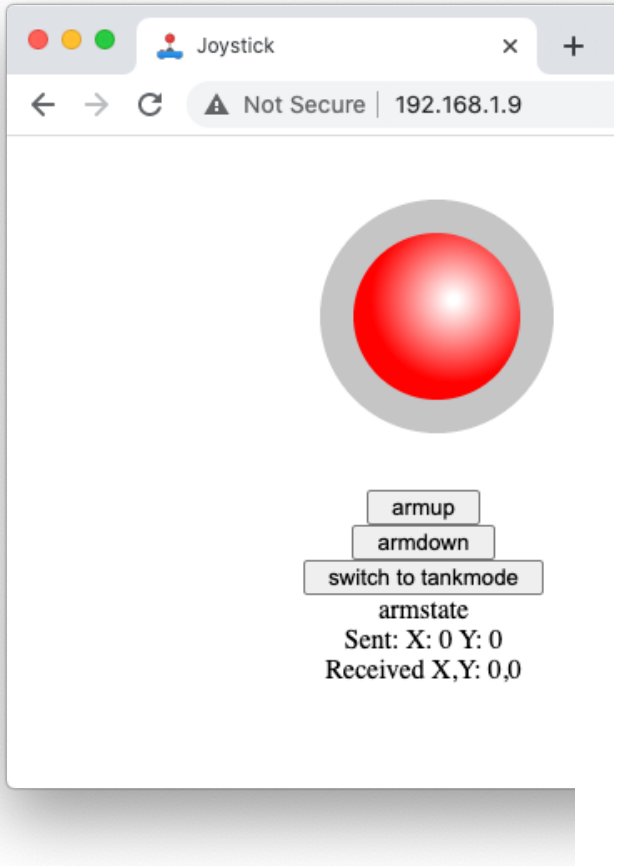

```
<body style="text-align: center;" >
  <canvas id="canvas" name="game"></canvas>
  <button type="button" onclick="armup()"> [...]
 [...]
```

HTML5 allows us to draw

```
// joystick UI
  window.addEventListener('load', () => {
      canvas = document.getElementById('canvas');
      ctx = canvas.getContext('2d');
      resize();

      document.addEventListener('mousedown', startDrawing);
      document.addEventListener('mouseup', stopDrawing);
      document.addEventListener('mousemove', Draw);

      document.addEventListener('touchstart', startDrawing);
      document.addEventListener('touchend', stopDrawing);
      document.addEventListener('touchcancel', stopDrawing);
      document.addEventListener('touchmove', Draw);
      window.addEventListener('resize', resize);

      document.getElementById("x_coord").innerText = 0;
      document.getElementById("y_coord").innerText = 0;
  });
```

Attach routines to events

joyJS.h




```
<script>
[...]
// limit transfer rate, 200 is 5Hz
  setInterval(myTimer, 200);
  var cts = 1; // clear to send
  function myTimer() {
    cts = 1;
    if (x_relative != joystate[0] || y_relative != joystate[1])
      upDate();
    }
  }

 function upDate(){
   joystick();
   sendJoy(x,y); // send x and y coord to ESP32
 }

function sendJoy(x,y) {
 [...]
  if (cts) {
    var xhttp = new XMLHttpRequest();
    var str = "joy?val=";
    var res = str.concat(x_relative,",",y_relative);
      [...readystate stuff not shown...]
    xhttp.open("GET", res, true);
    xhttp.send();
    cts = 0;
  }
}
```

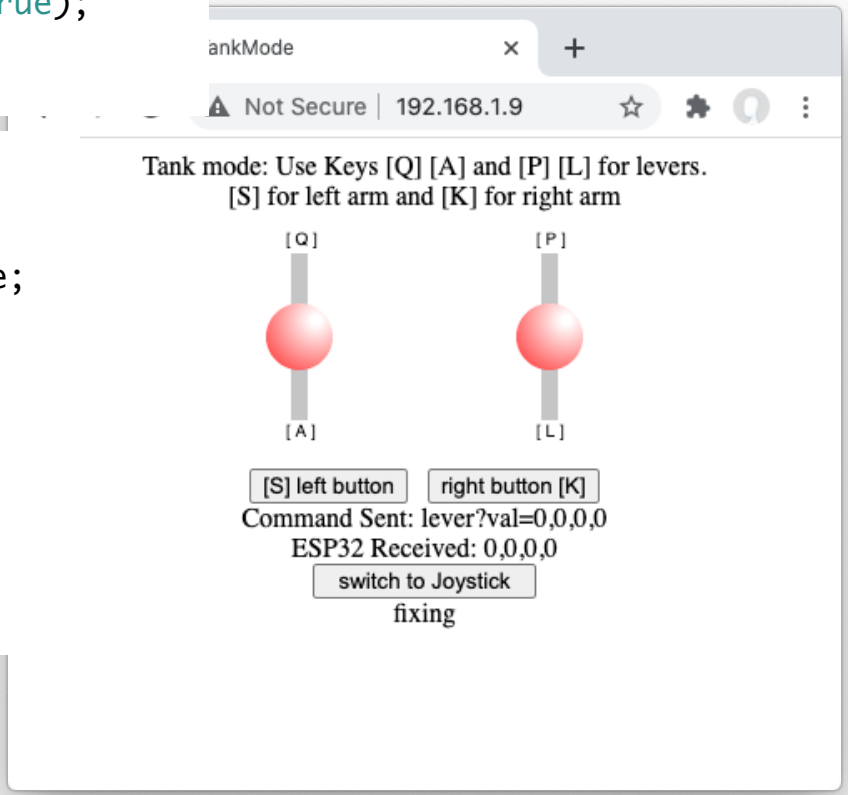
Periodically allow sender to send.

tankJS.h

```
// keyboard
  document.addEventListener('keydown', keyDownHandler, true);
  document.addEventListener('keyup', keyUpHandler, true);
  window.addEventListener('focus', focusChange);
  window.addEventListener('blur', focusChange);

  function keyDownHandler(event) {
    var code = keyboardCode(event);
      if (code == 0) return; // repeating
      document.getElementById("debug").innerHTML=code;
      if(code == 81 || code == 'q') { // Q key
         leftstate = -4;
      }
      if(code == 65 || code == 'a') { // A key
       leftstate = 4;
      }
       [... Other keys follow same format...]
      updateState();
  }
```

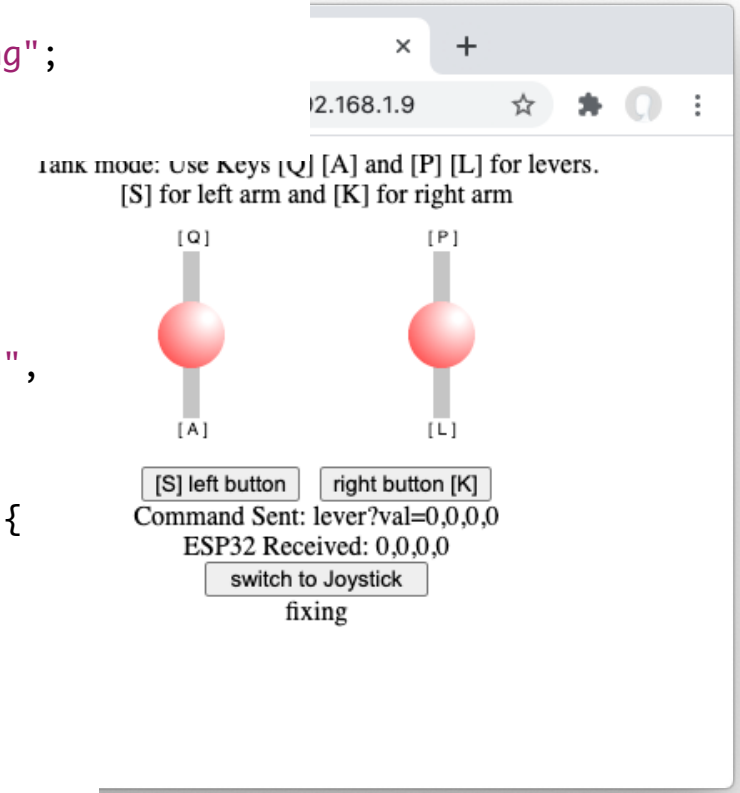
To find key mappings use https://keycode.info/

```
  function updateState(){
    drawLevers();
    sendState();
  }
```



```
// double check for inconsistency twice a second
  setInterval(myTimer, 500);
  function myTimer() {
    if (leftstate != leverstate[2] || rightstate != leverstate[3]) {
      sendState();
      document.getElementById("debug").innerHTML =  "fixing";
    }
  }

// UI functions via AJAX to esp32
  function sendState() {
    var xhttp = new XMLHttpRequest();
    var str = "lever?val=";
    var res = str.concat(leftarmstate,",",rightarmstate,",",
                         leftstate,",",rightstate);
    document.getElementById("levers").innerHTML = res;
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4) { if (this.status == 200) {
          resendflag = 0;
          leverstate = this.responseText.split(","); [...]
        }  else  resendflag = 1
      }
    };
    xhttp.open("GET", res, true);
    xhttp.send();
  }
```

tankJS.h



Lab4 Demo setup()

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  WiFi.begin(ssid, password);
  WiFi.config(IPAddress(192, 168, 1, 9),
        IPAddress(192, 168, 1, 1), IPAddress(255, 255, 255, 0));
  while(WiFi.status()!= WL_CONNECTED ) {
    delay(500); Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.printf("Use this URL http://%s/\n",WiFi.localIP().toString().c_str());
  server.begin();

  ledcSetup(RIGHT_CHANNEL0, SERVOFREQ, LEDC_RESOLUTION_BITS);
  ledcAttachPin(SERVOPIN1, RIGHT_CHANNEL0);
  ledcSetup(LEFT_CHANNEL1, SERVOFREQ, LEDC_RESOLUTION_BITS);
  ledcAttachPin(SERVOPIN2, LEFT_CHANNEL1);

  attachHandler("/joy?val=",handleJoy);
  attachHandler("/armup",handleArmup);
  attachHandler("/armdown",handleArmdown);
  attachHandler("/switchmode",handleSwitch);
  attachHandler("/lever?val=",handleLever);
  body = joybody;
  attachHandler("/favicon.ico",handleFavicon);
  attachHandler("/ ",handleRoot);
}
```

Old style server

Old style attach

The main joystick routine

The main tankmode routine

Lab4 Demo
Joystick mode

```
void handleJoy() {
  int left, right;
  x = getVal(); // from -50 to +50
  y = getVal();
  String s = String(x) + "," + String(y);

  left = x - y;
  right = x + y;

  leftservo = map(left, -50, 50, FULLBACK, FULLFRONT);
  rightservo = map(right, -50, 50, FULLBACK, FULLFRONT);

  sendplain(s);
  Serial.printf("received X,Y:=%d,%d\n",x,y);
}
```

Old style

Lab4 Demo
Tank mode

```
int leftstate, rightstate;
long lastLeverMs;

void handleLever() {
  leftarm = getVal();
  rightarm = getVal();
  leftstate = getVal();
  rightstate = getVal();
  String s = String(leftarm) + "," + String(rightarm) + "," +
             String(leftstate) + "," + String(rightstate);

  if (leftstate>0)      leftservo =  FULLBACK;
  else if (leftstate<0) leftservo =  FULLFRONT;
  else                  leftservo =  SERVOOFF;

  if (rightstate>0)      rightservo =  FULLFRONT;
  else if (rightstate<0) rightservo =  FULLBACK;
  else                   rightservo =  SERVOOFF;

  lastLeverMs = millis(); //timestamp command
  sendplain(s);
  Serial.printf("rec'd %d %d %d %d \n",leftarm, rightarm, leftstate, rightstate);
}
```

Old style

Lab 4 Demo Switching between modes:

tankJS.h

- string tankbody defined

joyJS.h

- string joybody defined

Old style

```
void handleSwitch() { // Switch between JOYSTICK and TANK mode
  String s="";
  static int toggle=0;
  if (toggle) body = joybody;
  else body = tankbody;
  toggle = !toggle;
  sendplain(s); //acknowledge
}
```

Old style

```
void handleRoot() {
  sendhtml(body);
}
```

Lab4 Demo loop()

```
void loop()
{
  static long lastWebCheck = millis();
  static long lastServoUpdate = millis();
  uint32_t ms;

  ms = millis();
  if (ms-lastWebCheck > 2){
    serve(server,body);
    lastWebCheck = ms;
  }
  if (ms-lastServoUpdate > 1000/SERVOFREQ) {
    updateServos();
    lastServoUpdate = ms;
  }
}
```

Old style serve

Q5: How many times per sec is serve() called?

Non-blocking

Update servos 60 times per sec

Non-blocking

Summary

- Use millis() to keep track of time and to schedule when you want to call routines periodically.
- You can use the joystick and tankmode interface for your Lab4 and build on it for the final project.
- We can add/change other simple javascript functionality if you need it.

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. Rules of the game

B. Using OpAmp Circuit

C. Wall-following and Beacon Behaviors