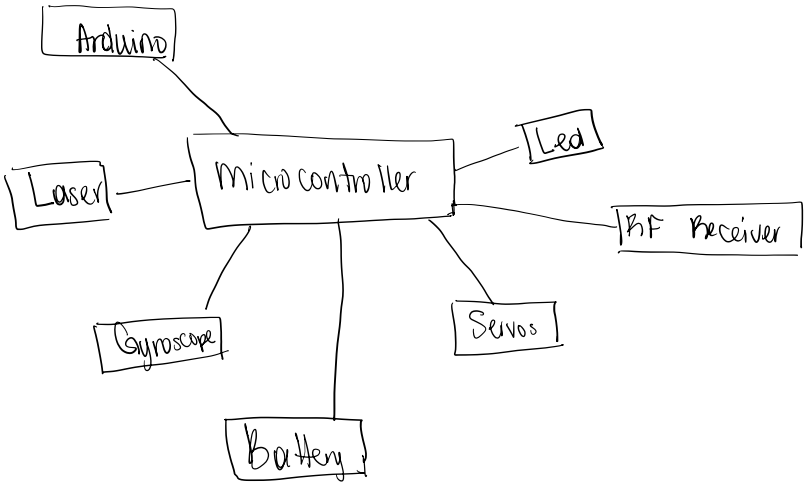
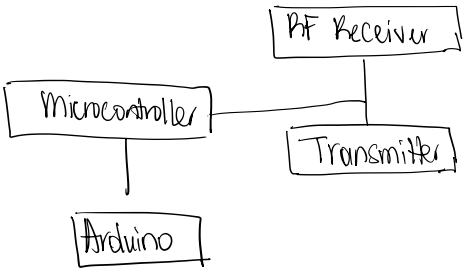
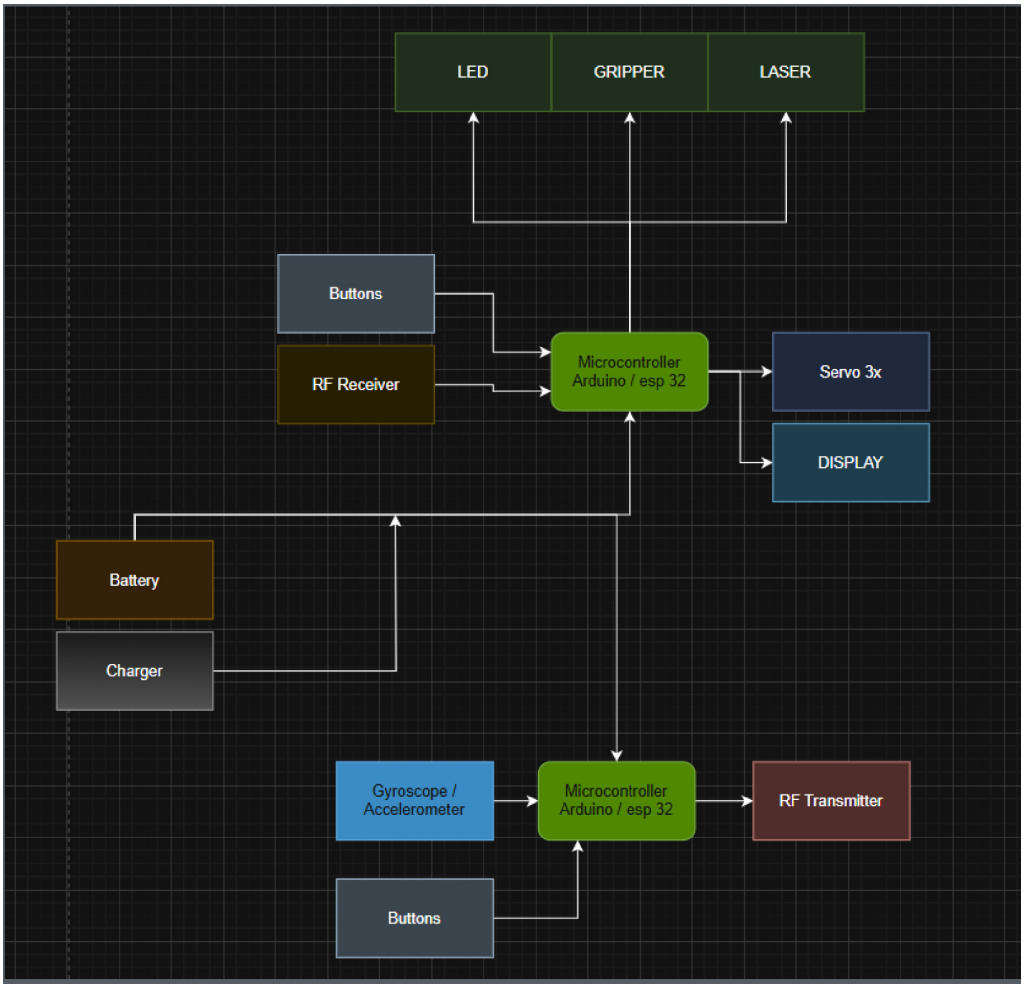
Block Diagram





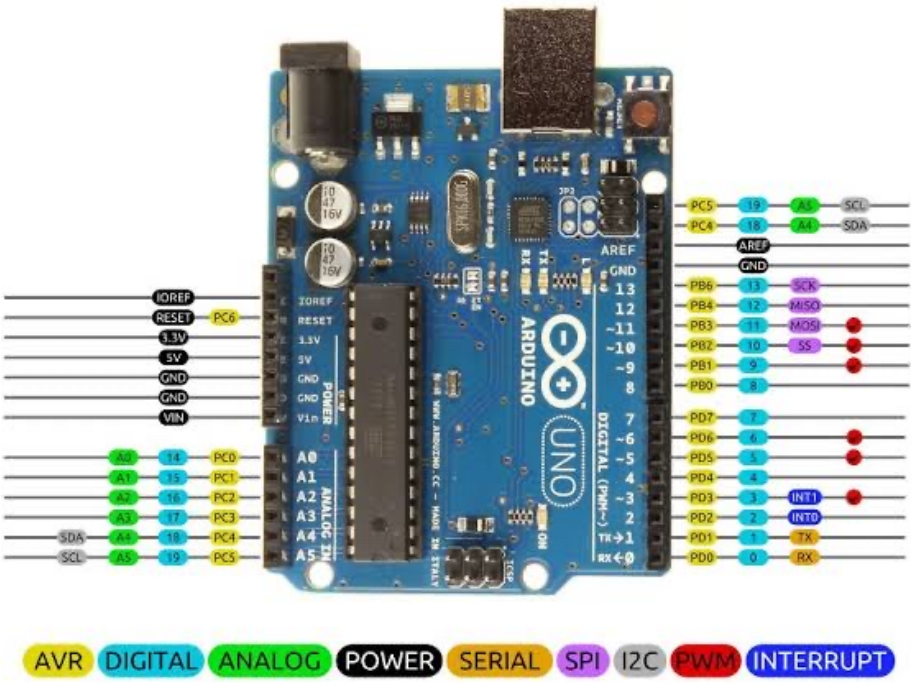


Proof of concept

Steps

* Wired first
  - Wireless
* Figure out the number of pins and board situation
*

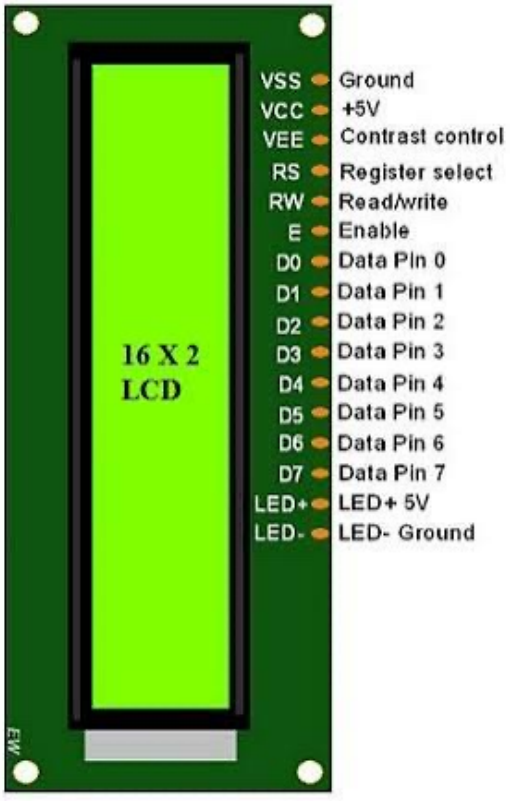
Arduino Uno R3 Pinout



Arduino

14 digital I/O

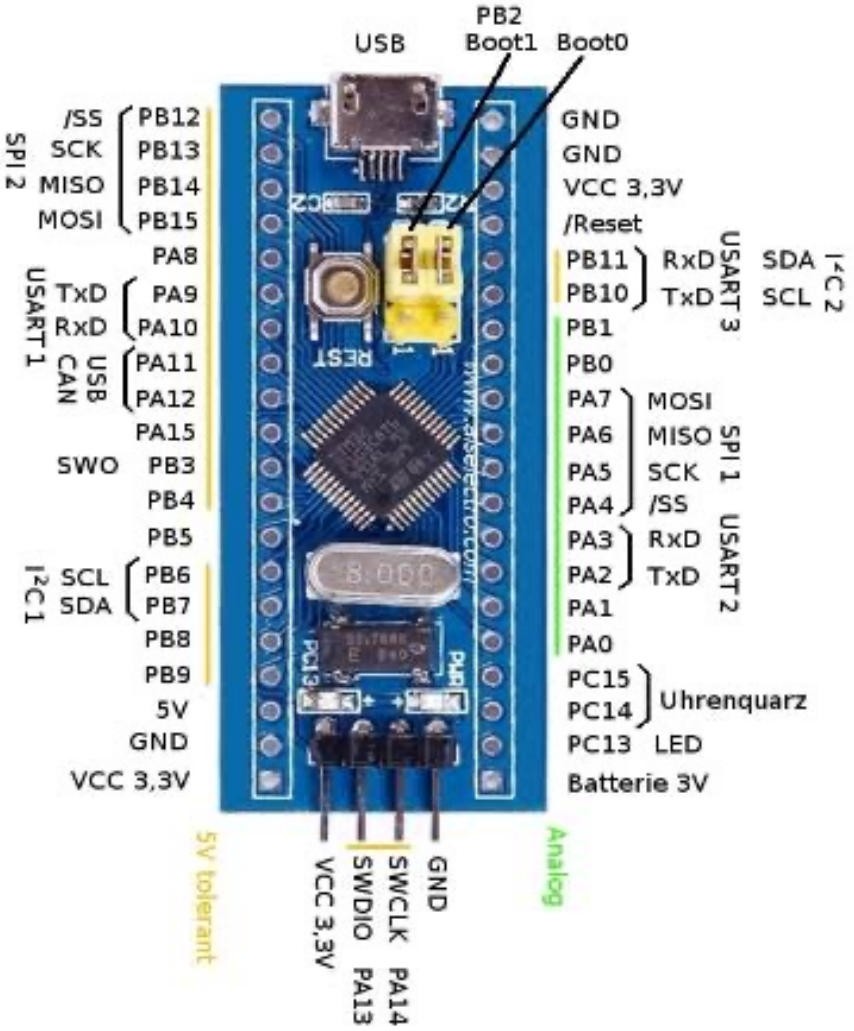
6 Analog (A0 - A5)



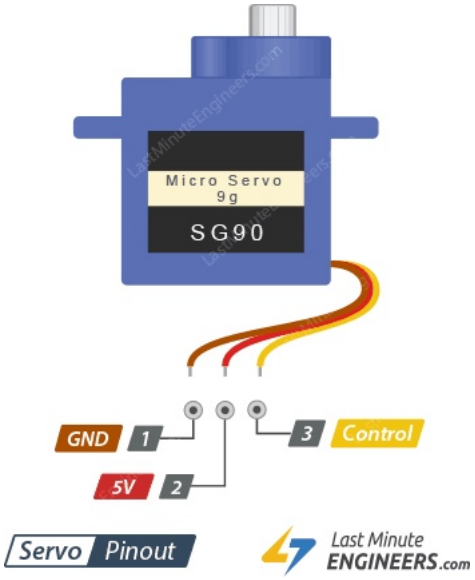
LCD Display

12 pins

STM32



37 GPIO pins



3 pins

Opt 1

2 Arduino

RF Receiver

Opt 2

Arduino
Esp 32 } modification for Ard so they can communicate

Opt 3

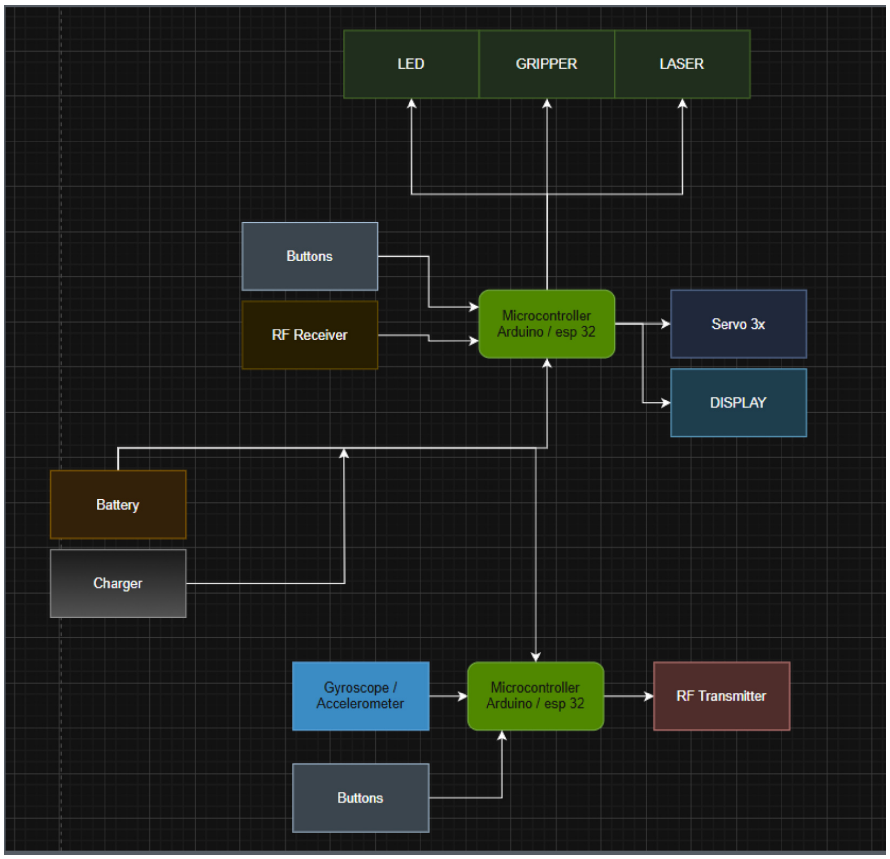
2 ESP32

* How you can draw schematics and do PCBs

9/18

- Figure out what prototypes we're doing
- EDA Options
  - Offline:
    - KiCAD Open Source
    - Cadence OrCAD, commercial
    - Altium Designer, commercial, very popular
    - Fritzing, Open Source, community, breadboards
    - Also; Eagle, DipTrace, LibrePCB
  - Online:
    - EasyEDA, linked at JLCP, LCSC, Cave

How to test the components



Components	Testing
1.) Buttons	Arduino
2.) RF Receiver	Arduino (x2)
3.) LED	

* Watch Strange parts video on PCB Fabrication

Inputs

push button options; B3FS or CK_KMS2xxG

Gyroscope; MPU-6050

RF Receiver; Conn_01x03_Mounting pin

Transmitter; Conn_01x03_Mounting pin

10/7

Development Environment

- IDE: Arduino
- Language: C++

Software Libraries

Standard Arduino

- Servo.h; This will simplify control of servo motors using PWM signals.
- LiquidCrystal.h; This will allow Arduino to interface with the LCD display.
- Wire.h; This will enable I^2C communication between Arduin and sensors.

RF Communication (Wireless Control)

- VirtualWire.h; Legacy RF library for simple transmit/receive
- RadioHead.h; This supports ASK, FSK and other RF protocols for modules like FS1000A (RF transmitter)
- RH_ASK.h (part of RadioHead)

- Motor Sensor (MPU6050 Gyroscope)
  - Adafruit_MPU6050.h; This handles raw gyroscope and accelerometer readings.
  - Adafruit_Sensor.h; Base class that standardizes data format
- Power and System Feedback
  - Adafruit_Battery.h; Used for monitoring battery LiPo battery life

10/9

1.) Troubleshooted Arduinos

2.) Downloaded the correct libraries

3.) Used Chatgpt to modify codes

MPU_9250

- Libraries Used;
  - Wire.h
  - mpu 9250
  - I2C
  - Bolder Flight System
- Pin out;
  - VCC
  - 3.3V
  - Gnd
  - SCL → A5
  - SDA → A4
  - EDA → ??
  - ECL → ??
  - ADD → ??
  - INT → ??
  - NCS → ??
  - FCSYNC → ??

To Do

* Figure out X, y, z orientation on the board
* Converting values to 180°
* Sending data to Servo

10/16

Checklist

- Proof of Concept Prototype
  - PCB design / layout
  - Datasheets
  - modules ordered
  - Download software libraries
  - Have development environment for software
  - Have breadboards, wires and mechanical elements
  - Have access to workbench with appropriate tools
- Determine meet time next week before presentation.

For schematic capture
- Add sheets
- Hierarchy view

10/17

Presentation Prep

- Software
- Libraries
- Code
- Speak on the MPU function, using your finger to create a motion that translates to the servo
    - Talk about the functions and libraries
    - Ask chatgpt to explain the functions for deeper understanding
- What would you use it for? (Research)

Presentation Topics

1) Introduction (Self)
   - Name
   - Major
   - Role
   - Form of communication
   -

2) Project Intro → John
   - What we're doing, How we're doing it, the goal
   - Material, Software, Hardware
     - → Explain MPU

3) Schematics; Explain Circuitry

4) KiCAD PCB: → Junior
   - Vias
   - Layer
   - Object placement

   * 3D model

5) Research → Junior
   - What you learned
   - Safety Precautions
   - How Hardware is implemented
   - Datasheets

6) Software → Alicia
   - Algorithm
   - Detail explanation of libraries

7) Performance / Demo
- User Story
- Demonstrate
- Call the audience to use it

8) Evaluations
- What we learning
- What was hard
- What we learned
- Experience (What your experience doing this project)

Question

- What would you use it for?

Application

- Lifting

Tasks

Alicia → Software, Testing the code (Study)

John → Testing the code, schematics, 3D model

Junior → Research, Powerpoint, PCB design (Routing, Vias, etc)

🎤 1-Minute Script: Software Explanation

> "For the software part of our project, we used Arduino C++ to integrate all of our components — the MPU6050 motion sensor, two servo motors, an LCD display, a push button, and an LED.

Software Overview (Smart Robotic Arm System)

The software for this project is designed around a **dual-mode control architecture** that allows the robotic arm to operate either **autonomously** or under **wireless remote control**.

The program is written in **Arduino C/C++** and is organized using a **state-based structure**. Two physical buttons are used to switch between operating modes in real time.

Inverse Kinematics is a mathematical process of figuring out joint angles for a connected system like ours

- Servo.h ; This will simplify control of servo motors using PWM signals.
- Liquid Crystal.h ; This will allow Arduino to interface with the LCD display.
- Wire.h ; This will enable I²C communication between Arduin and sensors.

- ServoTimer2 library: To control multiple hobby servo motors efficiently
- SR04.H ; for the ultrasonic sensor which measures distance by handling complex timing and calculations
- RH_ASK.h ; Radiohead library for enabling wireless communication using Amplitude Shift Keying (ASK)
- SPI.h ; communicates with serial peripheral interface (SPI) devices like SD cards, sensors or displays and acts like a controller. Uses dedicated pins (SCK, MOSI, MISO, SS)

- Motor Sensor (MPU6050 Gyroscope)
  - Adafruit_MPU6050.h ; This handles raw gyroscope and accelerometer readings.
  - Adafruit_Sensor.h ; Base class that standardizes data format
- Power and System Feedback
  - Adafruit_Battery.h ; Used for monitoring battery Life battery life

Tasks

* Research Kalman filter } Alicia
* Talk more about Libraries

* More research on safety precautions } Junior
* How implementation works

* 3D model
* PCB routing } John

10/31

- Equation for just one motor
  - Test it out without the MPU6050 and then add it afterwards

* 2 degree of freedom equation

L_1
L_2 >

$d = \sqrt{x^2 + y^2}$

∴ can reach target if $|L_1 - L_2| \leq d \leq L_1 + L_2$

- Finding θ_2

  Using Law of cosines for θ_2

$$\cos\theta_2 = \frac{(x^2 + y^2) - (L_1^2 - L_2^2)}{2L_1L_2}$$

- If $\cos\theta_2 > 1$, then its unreachable, so then:

$$\theta_2 = a\cos(\cos\theta_2)$$

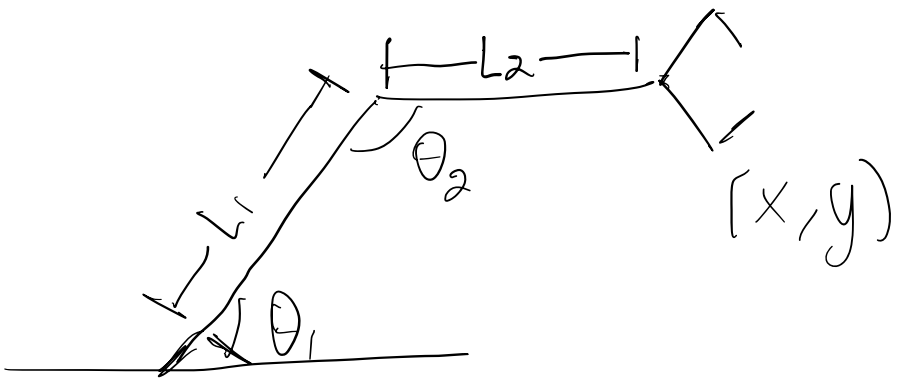
or $$\theta_2 = -a\cos(\cos\theta_2)$$

- Finding θ_1

Set $k_1 = L_1 + L_2\cos\theta_2$ and $k_2 = L_2\sin\theta_2$

$$\theta_1 = \text{atan2}(y, x) - \text{atan2}(k_2, k_1)$$

Visual Rep



1) 1-DoF sanity test (No MPU, just 1 servo)

— To test out the hardware/power

```
#include <Servo.h>
Servo s;
void setup(){ s.attach(3); }
void loop(){
  for (int a=0;a<=180;a+=2){ s.write(a); delay(15); }
  for (int a=180;a>=0;a-=2){ s.write(a); delay(15); }
}
```

2) 2-DoF IK test (type x, y in Serial, arm moves there)

— "math + servo" demo. No MPU yet.

```
#include <Servo.h>
#include <math.h>

Servo s1, s2;              // shoulder, elbow
const int PIN1=3, PIN2=5;     // adjust to your wiring
const float L1=100.0, L2=90.0; // mm (measure your links!)

float deg(float rad){ return rad*180.0/PI; }
float rad(float d){ return d*PI/180.0; }

// mount offsets/inversions per your horn orientation:
const float S1_OFFSET = 90.0;   // servo neutral angle for θ1=0
const float S2_OFFSET = 90.0;   // servo neutral angle for θ2=0
const int   S1_DIR = +1;        // flip sign if motion is reversed
const int   S2_DIR = +1;

void setup(){
  Serial.begin(115200);
  s1.attach(PIN1);
  s2.attach(PIN2);
  Serial.println(F("Enter target as: x y (mm). Example: 120 40"));
}
```

```
 void moveToXY(float x, float y){
  float d2 = x*x + y*y;
  float c2 = (d2 - L1*L1 - L2*L2)/(2*L1*L2);
  c2 = constrain(c2, -1.0, 1.0);             // numeric safety

  // choose elbow-down (+acos) or elbow-up (-acos)
  float th2 = acos(c2);                      // radians (elbow-down)
  // float th2 = -acos(c2);                   // use this for elbow-up

  float k1 = L1 + L2*cos(th2);
  float k2 = L2*sin(th2);
  float th1 = atan2(y,x) - atan2(k2,k1);       // radians

  // convert to servo angles
  float a1 = S1_OFFSET + S1_DIR*deg(th1);
  float a2 = S2_OFFSET + S2_DIR*deg(th2);

  // joint limits (adjust to your mechanics)
  a1 = constrain(a1, 0, 180);
  a2 = constrain(a2, 0, 180);

  s1.write((int)a1);
  s2.write((int)a2);

  Serial.print(F("θ1=")); Serial.print(deg(th1),1);
  Serial.print(F("°, θ2=")); Serial.print(deg(th2),1);
  Serial.print(F("°  │  S1=")); Serial.print(a1,0);
  Serial.print(F("  S2=")); Serial.println(a2,0);
}

void loop(){
  if (Serial.available()){
    float x,y;
    x = Serial.parseFloat();
    y = Serial.parseFloat();
    if (Serial.read()=='\n'); // eat newline
    float d = sqrt(x*x+y*y);
    if (d < fabs(L1-L2) || d > (L1+L2)) {
```

```
      Serial.println(F("Unreachable: violates |L1-L2| <= d <= L1+L2"));
    } else {
      moveToXY(x,y);
    }
  }
}
```

3) MPU 6050

- Map roll/pitch to polar target (r, ϕ), then convert to (x, y) and then run the same IK

∴ KalmanAngle Roll / Pitch (compute)

```
// Map sensor to a polar target (tune ranges to taste)
float phi   = constrain(KalmanAngleRoll,  -60, 60) * PI/180.0;        // direction
float rMin=40, rMax=L1+L2-10;  // keep a little margin
float r = map((long)KalmanAnglePitch, -30, 30, rMin, rMax);           // reach
r = constrain(r, rMin, rMax);

// Convert to x,y and call IK
float x = r * cos(phi);
float y = r * sin(phi);
moveToXY(x,y);
```

10/31

Zoom call with Group members

Math functions | code → Alicia

Research → Junior

Tested out two inverse kinematic equations with servo → John

11/1

Implemented potentiometer to give us X and y values
↳ Angle for the servos

To do
- To get tilt value from the MPU6050 and convert it into X and Y values instead of using potentiometer

11/7

* Testing out the inverse kinematics for two motors
* Design Decision
  - 5 DOF

11/16

5 DOF

Joint	Motion Type	DoF Contribution
Base	rotation(left/right)	1
Shoulder	Pitch	1
Elbow	Pitch	1
Wrist Pitch	Pitch	1
Wrist roll	Roll(rotation)	1

3 DoF affecting X,Y,Z position
(Shoulder, elbow, base)

2 DoF affecting orientation
(wrist pitch, wrist roll)

Inverse Kinematics

- **Position IK (first 3 motors only)**

Compute (Base Yaw, Shoulder Angle, Elbow Angle), given the target X, Y, Z

_Base rotation (only aims the arm horizontally)

$$\theta_{base} = \text{atan2}(y, x)$$

3 DoF system positions the end of the arm at the desired point in 3D space using the geometry of a two-link arm

_Solve the 2-link planar IK

Projecting the target into the plane of the arm

$$r = \sqrt{x^2 + y^2}$$

$$z = z_{target}$$

reach condition

$$|L_1 - L_2| \leq d \leq L_1 + L_2$$

L_1 (upper arm), L_2 (forearm)

$$d = \sqrt{r^2 + z^2}$$

_Elbow Angle

$$\cos\theta_2 = \frac{d^2 - L_1^2 - L_2^2}{2 L_1 L_2}$$

$$\theta_2 = \arccos(\cos\theta_2)$$

Find theta first and plug it into sin

_Shoulder Angle

$$\theta_1 = \text{atan2}(z, r) - \text{atan2}(L_2 \sin\theta_2, L_1 + L_2 \cos\theta_2)$$

- **Orientation IK (last 2 motors)**

* Wrist pitch aligns the gripper up/down
* Wrist roll rotates the gripper

2 DoF system that adjusts the wrist orientation without changing the endpoint position

Once the shoulder and elbow angles are known

$$\theta_{wrist\,pitch} = desiredPitch - (\theta_1 + \theta_2)$$

$$\theta_{wrist\,roll} = desiredRoll$$

We then compute the base rotation with atan(y, x). Then we use the law of cosines to calculate the elbow angle and atan2 geometry to calculate the shoulder angle

* We must use an external 5V supply in order to power 5 motors

Code

```
#include <Servo.h>

Servo baseServo;
Servo shoulderServo;
Servo elbowServo;
Servo wristPitchServo;
Servo wristRollServo;

void setup() {
  baseServo.attach(3);       // set your pins!
  shoulderServo.attach(5);
  elbowServo.attach(6);
  wristPitchServo.attach(9);
  wristRollServo.attach(10);

  Serial.begin(115200);
  Serial.println("5-DOF Servo Test Ready");
}

void sweepServo(Servo &s) {
  for (int pos = 0; pos <= 180; pos+=2) {
    s.write(pos);
    delay(10);
  }
  for (int pos = 180; pos >= 0; pos-=2) {
    s.write(pos);
    delay(10);
  }
}

void loop() {
  Serial.println("Base Sweep");
  sweepServo(baseServo);

  Serial.println("Shoulder Sweep");
  sweepServo(shoulderServo);
```

```
    Serial.println("Elbow Sweep");
    sweepServo(elbowServo);

    Serial.println("Wrist Pitch Sweep");
    sweepServo(wristPitchServo);

    Serial.println("Wrist Roll Sweep");
    sweepServo(wristRollServo);

    delay(1000);
}
```

11/18

5 DoF Testing

- Originally, the code showed us how each servo moved. They basically moved one at a time without our control. All 5 would move and start the process over and over.
- Then, we modified the code to where we can control the servos individually through the Serial monitor.

Commands:

b = base to angle	b = 90
s = shoulder to angle	s = 45
e = elbow to angle	e = 90
p = wrist pitch to angle	p = 90
g = gripper (safe limited)	g = 90

- Now we want to set limits and apply them to the code.

12/2

- Isaac Sim LAB
- Test the code on robot
- Make breadboard
  - ↳ Transfer to PCB
- Testing the limits on the robot
- Slides

12/4

Study the code

- HCPCA9685 library for the Servo Driver
- Initial parking position value of the motor (Cons int = 60)
- Degree of robot servo sensitivity - Intervals

* To keep track of the current value of the motor positions

12/9

- Tested the transmitter and receiver

To do

- LCD Display
- Perfect the MPU6050

Challenges

- Radiohead and servo.h libraries uses the same internal timer on the Arduino Uno. So we might have to switch to Arduino Mega or use wifi / bluetooth.

* Transmitter
  - Pin 12

- Wiring PCB
- Documentation (Final Report)
- We need to get an I2C LCD
- Arduino Mega
- We need to implement the code into the transmitter

Testing – Robot Code

- Code that controls the robot without the MPU6050.
- The way the code works is by sending a command through the serial monitor i.e if the SM reads a character, the base rotates right or left

  L means base rotates by an increment ± 10

- Transmitter sends characters (L, R, U, etc) to the receiver and then the Arduino with the receiver will command the servo to increase its angle by ±.

Testing – Transmitter & Receiver

- Lele found a guy on Youtube and tested his code. But then we realized that our antenna was too short and we couldn't receive a signal but we fixed it and now it works.

  → DroneBot Workshop (Youtube Channel)

Tasks

- Research on how RadioHead and Servo.h libraries can work together:
- LCD Display, buttons, joystick
- PCB

→ * You can force the RadioHead library to use Timer 2 instead. This can be done by enabling the define RH_ASK_ARDUINO_USE_TIMER2 near the top of RH_ASK.cpp

12/12

Testing Issues

- We are having issues with the transmitter and receiver

12/19

Updated Block Diagram