Final Report Group #22

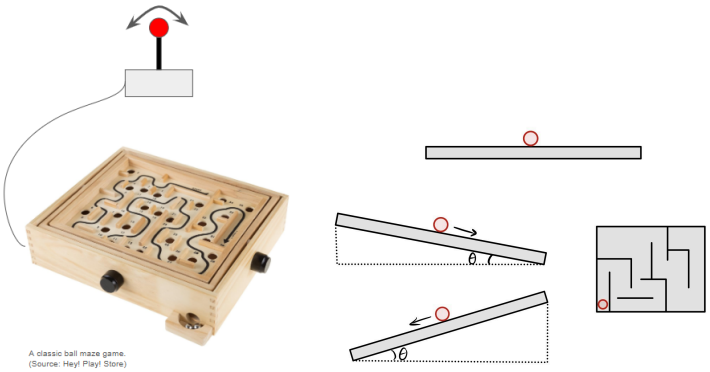
Jonathon Cheng, Sacha Dulout, Jaden Nguyen and Ole G. F. Sjulstad
December 19, 2024

Opportunity

To alleviate boredom and promote emotional well-being, the aim was to create a physical (non-digital) entertainment solution. Our goal is to provide an engaging experience that doesn't rely on phones, tablets, or computer screens, making it ideal for moments of solitude. This led us to the idea of a classic ball maze game, but controlled with a joystick through two DC motors. It is a tactile and interactive game designed to captivate users while offering a refreshing break from digital distractions.

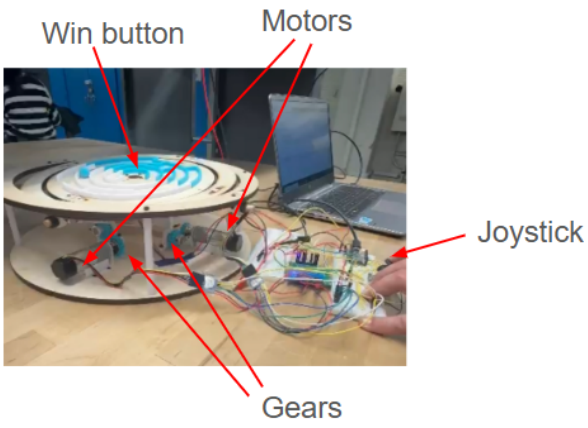
High-level strategy

The user moves a joystick to generate input signals, which are processed by a proportional controller implemented within an ESP32. The controller maps the joystick's position to a corresponding motor position, allowing the motors to adjust the angles of a board/maze by tilting it along two axes. By changing the board's slope, the user can guide a ball through a maze with precision.



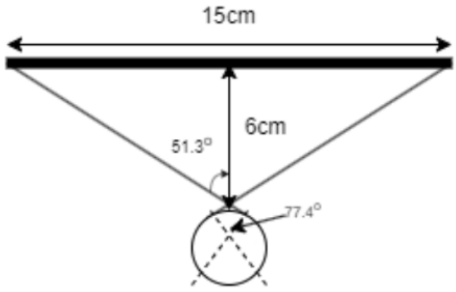
A classic ball maze game.
(Source: Hey! Play! Store)

Integrated Physical device



Function-critical decisions

The critical part of our design was that the tilting of the maze is done using a cable wrapped around a shaft, one for each axis, making two shaft in total. These shafts are driven by motors and the rotation of the shafts puts tension on the cable which tilts the maze plate. The mechanical connection between the shaft and the motor is performed by spur gears (gear ratio 1:1).



Given a torque of around 7.31Ncm on an 8mm shaft, the tension, T :

$$\tau = T \cdot r_{shaft} \Rightarrow T = \frac{\tau}{r_{shaft}}$$

$$\Rightarrow \frac{0.0731}{0.004} = 18.275N$$

Through simple trigonometry, for the most loaded scenario, the angle at which the cable exits towards the platform is around 51.3°, which in turn provides an angle of 77.4° at the capstan.

Assuming the frictional coefficient to be around 0.2 between the cable and the shaft, and that there will be a total of 3 full wraps and 78.5% of a wrap (due to the capstan angle):

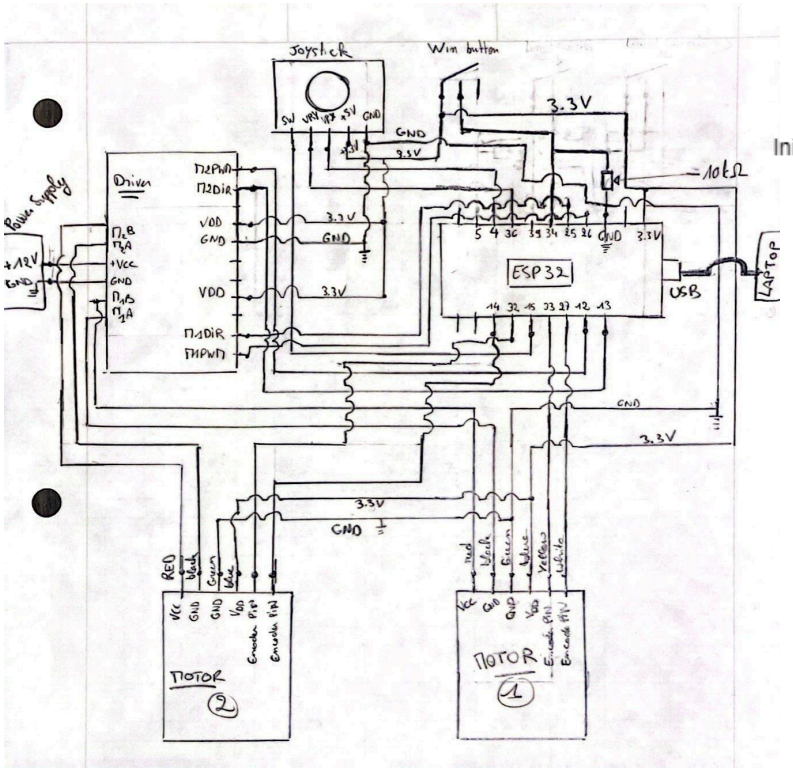
$$\phi = 3(2\pi) + 0.785(2\pi) = 9.57\pi$$

$$T_1 = \frac{T_2}{e^{\mu\phi}} = \frac{18.275}{e^{(0.1)(7.57\pi)}} = 1.69N \approx 2N$$

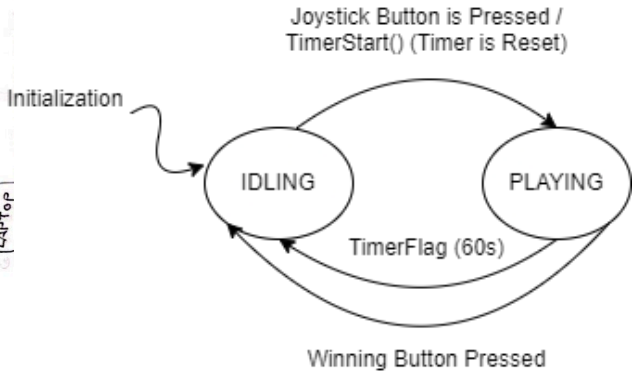
A preload of approximately 2N is needed for the cable which has been wrapped fully three times.

Circuit diagram and State Transition Diagram

Circuit diagram:



State transition diagram



Reflections for future students

Throughout this semester we've worked consistently on our project by setting weekly meetings. Having a meeting each week allowed us to set realistic goals, give each other feedback, and make design decisions together. If we felt like we didn't have anything to discuss or if other classes had to be prioritized, we our group was understanding of this. We've each had our roles to play and played to each other's strengths and weaknesses. We're overall happy with the progress we've made over the course of the semester, but there will always be aspects one can improve. For us, some of these are making the tensioning of the strings easier to adjust, making different designs for the maze (both easier and more challenging), and compacting our wiring.

If anything, to sum up our biggest takeaways for future students:

- Start early with a small-scale prototype in case you need to pivot, make bigger changes, etc.
- Don't procrastinate, get on with it
- Weekly meetups have been invaluable for us. This doesn't necessarily mean that others have to take this approach, but find something that works for you and stick with it.
- Remember to have fun!

Appendices

Appendix: Bill of Materials

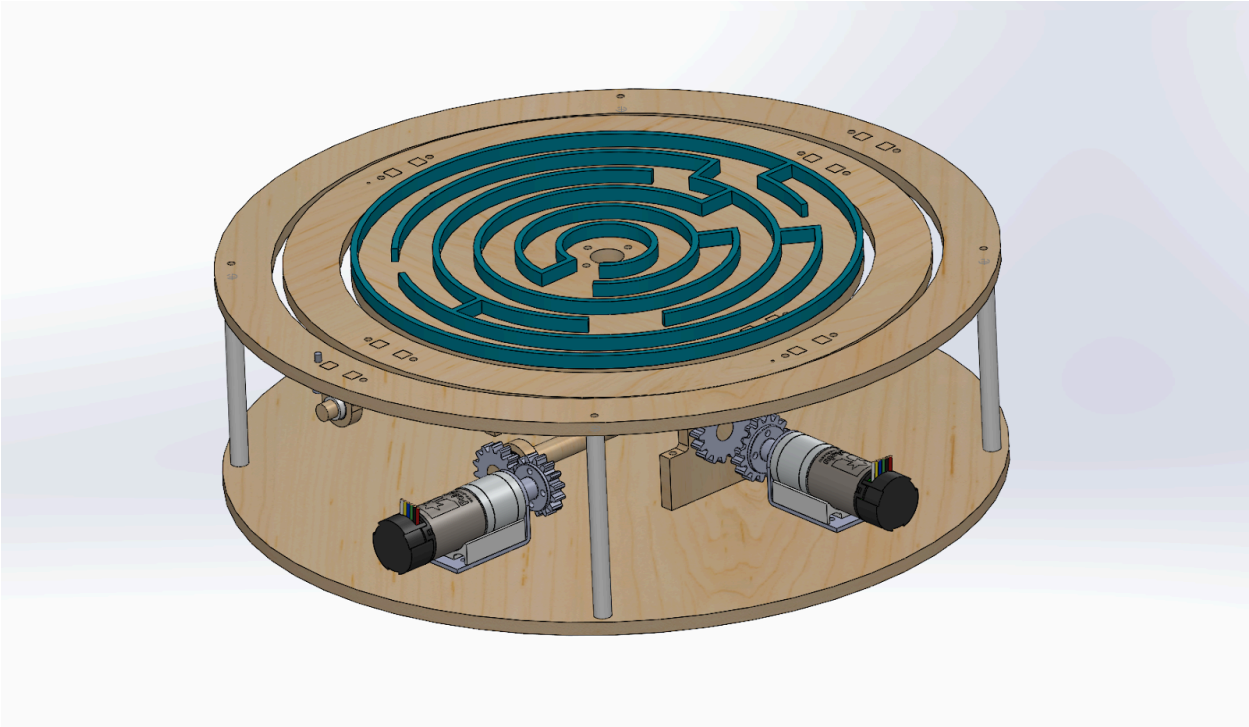
FULL BILL OF MATERIALS								
ITEM NO.	**DESCRIPTION**	**COMPANY**	**PART NO.**	**LINK**	**PRICE / UNIT**	**COUNT**	**TOTAL PRICE**	**NOTES**
1	Rotary Shaft - 303 Stainless Steel 8mm x 400mm	McMaster-Carr	4143N19	https://www.mcmaster.com/products/shafts/diameter~8-mm/shafts-2~/	$23.15	2	$46.30	
2	Plywood 1/4"- 2' by 4'	Home Depot	Internet # 202093790 Model # 103064 Store SKU # 103064	https://www.homedepot.com/p/Handprint-1-4-in-x-2-ft-x-4-ft-Sande-Plywood-Project-Panel-103064/202093790	$14.94	2	$29.88	
3	75:1 Metal Gearmotor 25Dx69L mm HP 12V with 48 CPR Encoder	Pololu	4752	https://www.pololu.com/product/4846	$48.95	2	$97.90	
4	Pololu Dual MC33926 Motor Driver Shield for Arduino	Pololu	2503	https://www.pololu.com/product/2503	$36.95	1	$36.95	
5	Analog 2-axis Thumb Joystick with Select Button + Breakout Board	Adafruit	512	https://www.adafruit.com/product/512	$5.95	1	$0	
6	Assembled Adafruit HUZZAH32 – ESP32 Feather Board - with Stacking Headers	Adafruit	3619	https://www.adafruit.com/product/3619	$21.95	1	$0	Part of the Microkit
7	Low-Stretch Easy-Splice Rope—For Lifting	McMaster-Carr	2222T11	https://www.mcmaster.com/products/rope/rope-2~/low-stretch-easy-splice-rope-for-lifting/	$0.72	5	$3.60	

8	8mm diameter bearings	Amazon	MR128 2RS	https://www.amazon.com/dp/B082PPLHLQ?ref=ppx_yo2ov_dt_b_fed_asin_title	$0.95	4	$3.80	
9	10mm diameter bearings	Amazon	6700ZZ	https://www.amazon.com/uxcell-6700ZZ-Groove-Bearings-Shielded/dp/B082PPYZQX/ref=sr_1_3?crid=LM7Z66KVQ1SW&dib=eyJ2IjoiMSJ9.AcXdUGWCV4qu091HuxEQg9LN5eMZyBLxRujDgI5onU5-5HUOFWmmWDcJBJKGazhM8_EZbaozTaTCKEDAvrT4xGTaHo7EdP1uV6udrV0NVg3mwZdog6XWJMvs5OouqvHAgTQeAMcgEEun9jQGellESymPl2C4OQbCvB7vCE0U8BPQ8TKN7S0FxN5REeVDUS0brqCpQpbj9gbR5PtL3jpl_oDhTdF8eQSqmE3ihf_t18g.upa6qc1VYUE1ixPbwGMuknZK9FPfC0A7DmDhsgZBgko&dib_tag=se&keywords=10mm%2Bbearing&qid=1729291260&sprefix=10mm%2Bbearing%2Caps%2C183&sr=8-3&th=1	$0.76	4	$3.04	
10	M4 screws	McMaster	90380A374	https://www.mcmaster.com/products/screws/tapping-screws~/length~6-mm/drive-style~phillips/	$0.09	30	$2.70	
11	Cup / Eye hook (to fix the cable)	Amazon	N/A	https://www.amazon.com/Cyful-Ceiling-Self-Tapping-Screws-Golden/dp/B07RKS9Q88/ref=sr_1_8?crid=1QSCNNFDGJUBW&dib=eyJ2IjoiMSJ9.Gwo4Qch7xfVLLedWF85uidny6j3TUvfiAXPCkn00b20LLhoRgRKNZl4ByNtFqq-DdyfO-pQo3toLlUQFi7Wz2DDwA23-VeFWgSyGwnaeWOlrAUE1HuBh8fo_ZC6TMwo3dZnZnk2enkvk_j9kRNJ_hrZRc_fKXjccMPW268-QGhTtyJYtnqPw0bHrkTCNksZt2kCYdYyjpofrTu4tLEpvW0HR3iIjPaj_EPYdfvhDwSQ.Im1EcoZdRMv9W7uVWtG5i2jMYdJzdVZSSRivtDa-L80&dib_tag=se&keywords=cup%2Bhook&qid=1729480405&sprefix=cup%2Bhook%2Caps%2C342&sr=8-8&th=1	$0.06	4	$0.24	

12	Spring (for preload)	Amazon	N/A	https://www.amazon.com/uxcell-Extended-Compressed-Diameter-Stainless/dp/B07NRT3P8Z/ref=sr_1_1_sspa?crid=3NI7JV2R6XYTT&dib=eyJ2IjoiMSJ9.a4eBq687FXwcBIYQtZzif0d6ZNAN73Zp9BnqGpL6G-0G6KJYsfmG4B2LjFrtR3j5GWEj-JKDHTXQ3bduMgCZN-LfT9RWH-Q70WIlOIr7rRoOf5wt-cr-E7MLaL4JZnHJJ9Hzj7YddUilVLxeEgXAl127hQmwqPQbg4M07s50QZ9qOMtvxuf4Q7DMNnceTnCy1MgpG9rrCfBf-Eozkoy4gDIrphhI_l52v_DjnD-8fuI.QdiOZaxGh2IhXz1uVYYc7neQAuoj7ZYaSu5oahuoWDQ&dib_tag=se&keywords=small%2Btension%2Bsprings&qid=1729483044&sprefix=small%2Btension%2Bsprings%2Caps%2C245&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&th=1	$0.88	4	$3.52	
13	Shaft clamp 8mm diameter	McMaster	7731N29	https://www.mcmaster.com/products/clamp-collars/clamping-two-piece-shaft-collars-9/system-of-measurement~metric/shaft-diameter~8-mm/width~5-mm/od~20-mm/material~2024-aluminum/	$16.20	2	$32.40	
14	Shaft clamp 10mm diameter	McMaster	7731N23	https://www.mcmaster.com/products/clamp-collars/clamping-two-piece-shaft-collars-9/system-of-measurement~metric/shaft-diameter~10-mm/width~5-mm/od~20-mm/clamping-screw-material~steel-1/	$16.20	2	$32.40	
15	Shim 8mm diameter	McMaster	99432A343	https://www.mcmaster.com/products/shims/shims-2~/system-of-measurement~metric/id~8-000-mm/id~8-0-mm/id~8-mm/od~10-000-mm/od~10-0-mm/od~10-mm/	$0.50	4	$2.00	
16	Shim 10mm diameter	McMaster	92927A574	https://www.mcmaster.com/products/shims/shims-2~/system-of-measurement~metric/id~10-000-mm/id~10-0-mm/id~10-mm/od~14-73-mm/	$1.50	4	$6.00	
17	Belleville washer 8mm diameter	McMaster	91235A616	https://www.mcmaster.com/products/belleville-washers/lock-washers~/washer-type~conical/system-of-measurement~metric/screw-size~m8/id~8-2-mm/	$1.20	2	$2.40	
18	Belleville washer 10mm diameter	McMaster	90407A112	https://www.mcmaster.com/products/belleville-washers/system-of-measurement~metric/id~10-000-mm/thickness~0-800-mm/thickness~0-8-mm/	$3.20	2	$6.40	

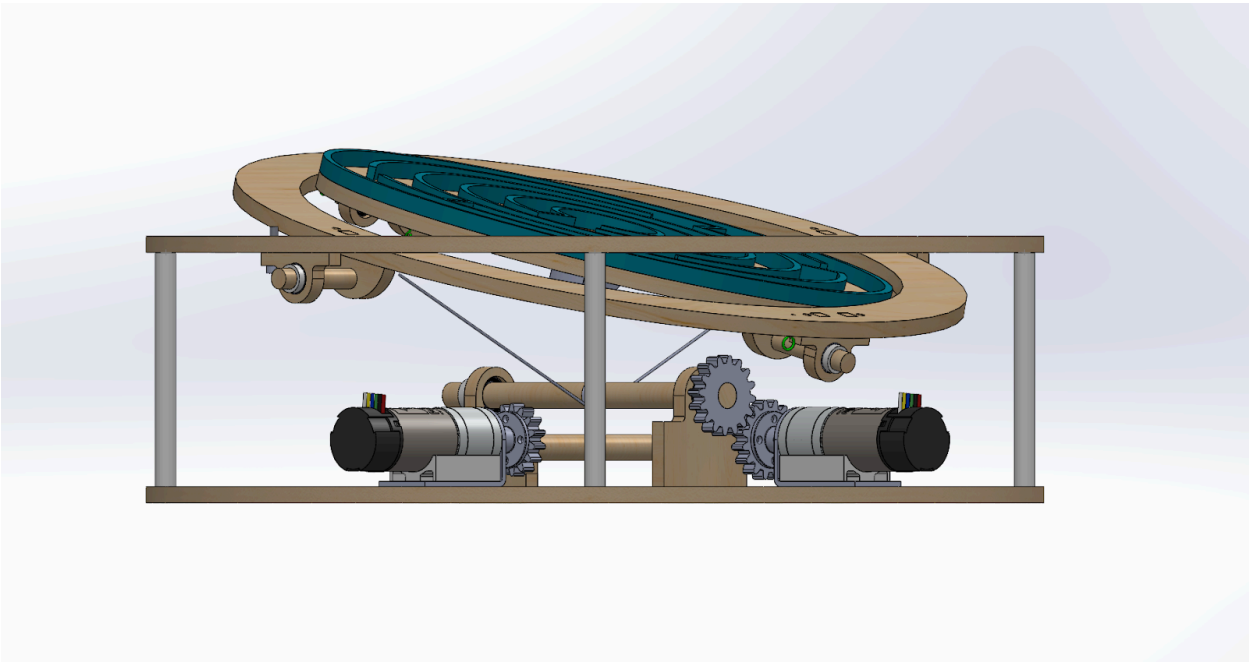
19	Wooden shaft 8mm diameter	Amazon	N/A	https://www.amazon.com/dp/B0BHT3YZRH?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1	$0.41	2	$0.82	
						Total	**$310.35**	

Appendix: Images of the CAD model

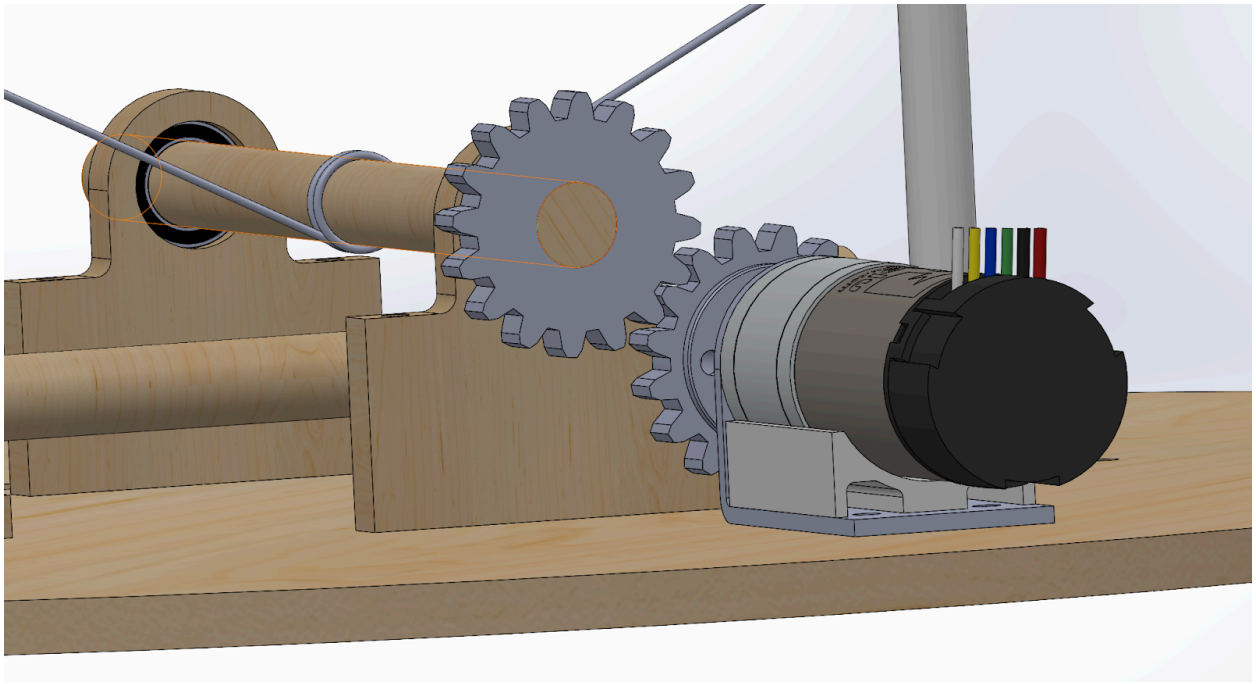


Isometric View

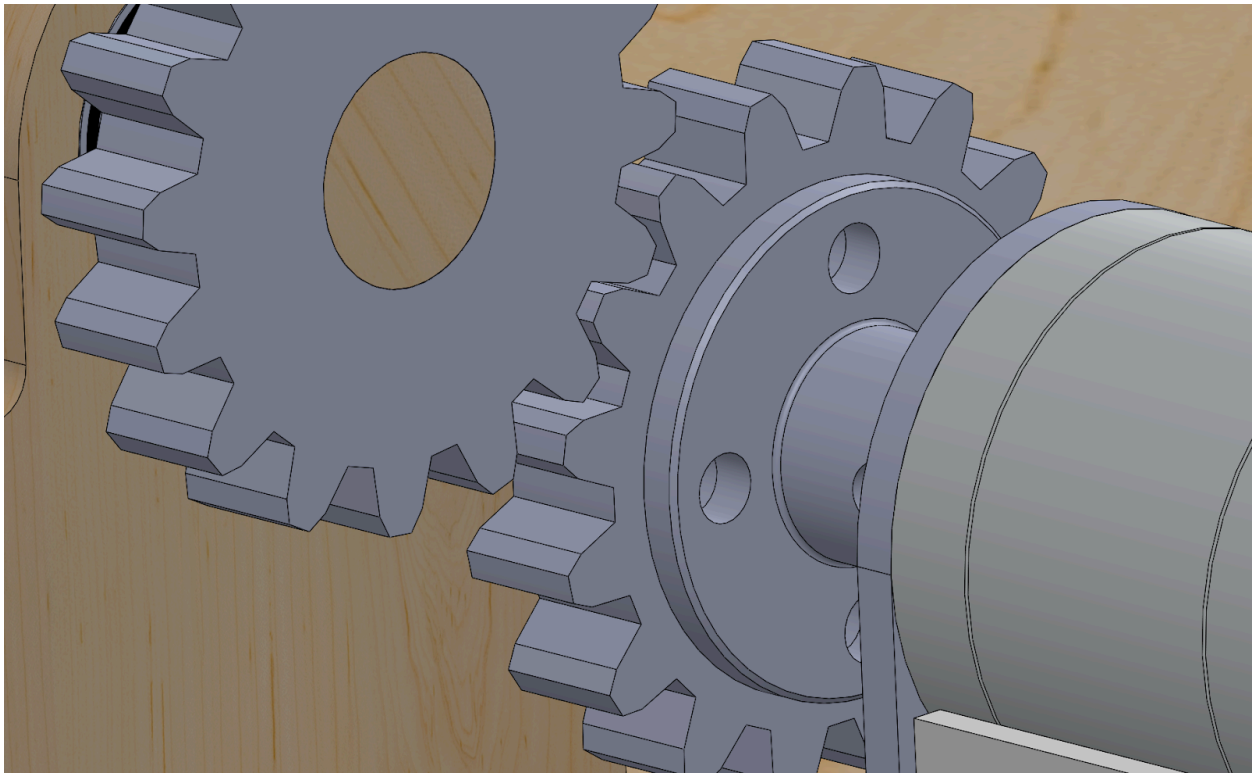
Top view



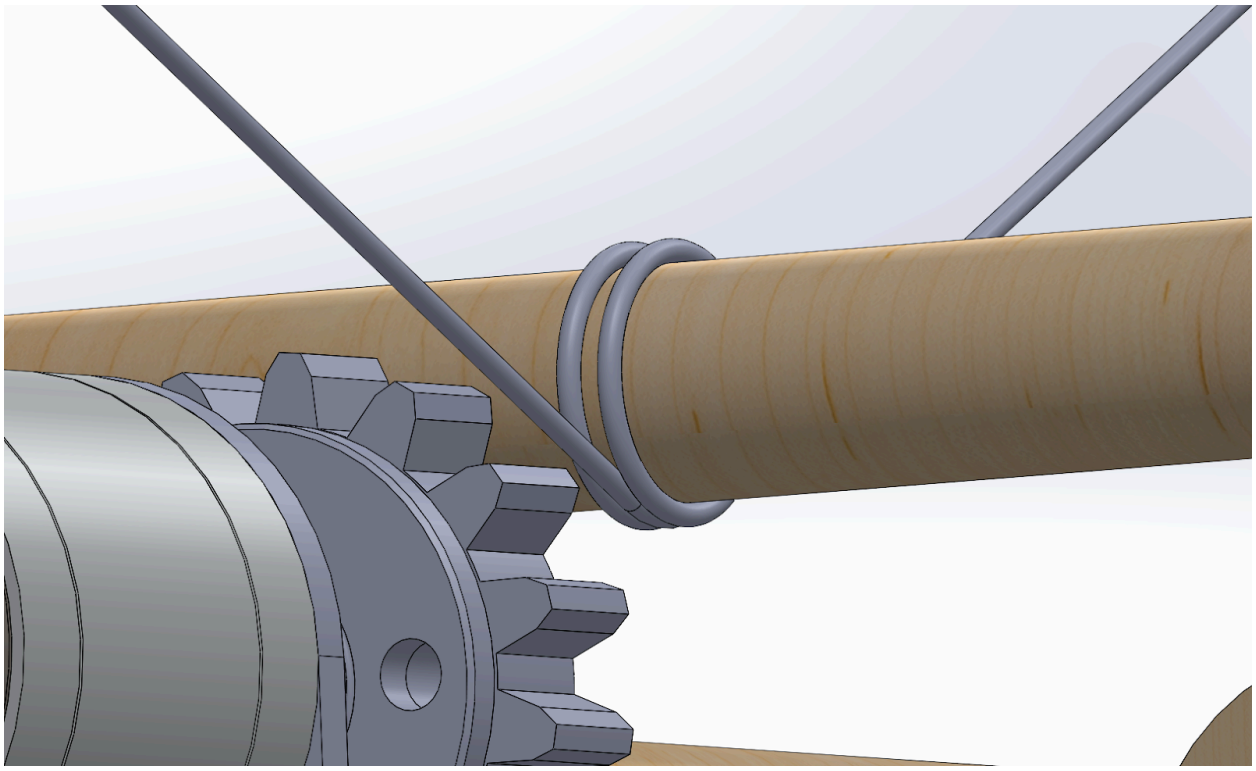
Side view



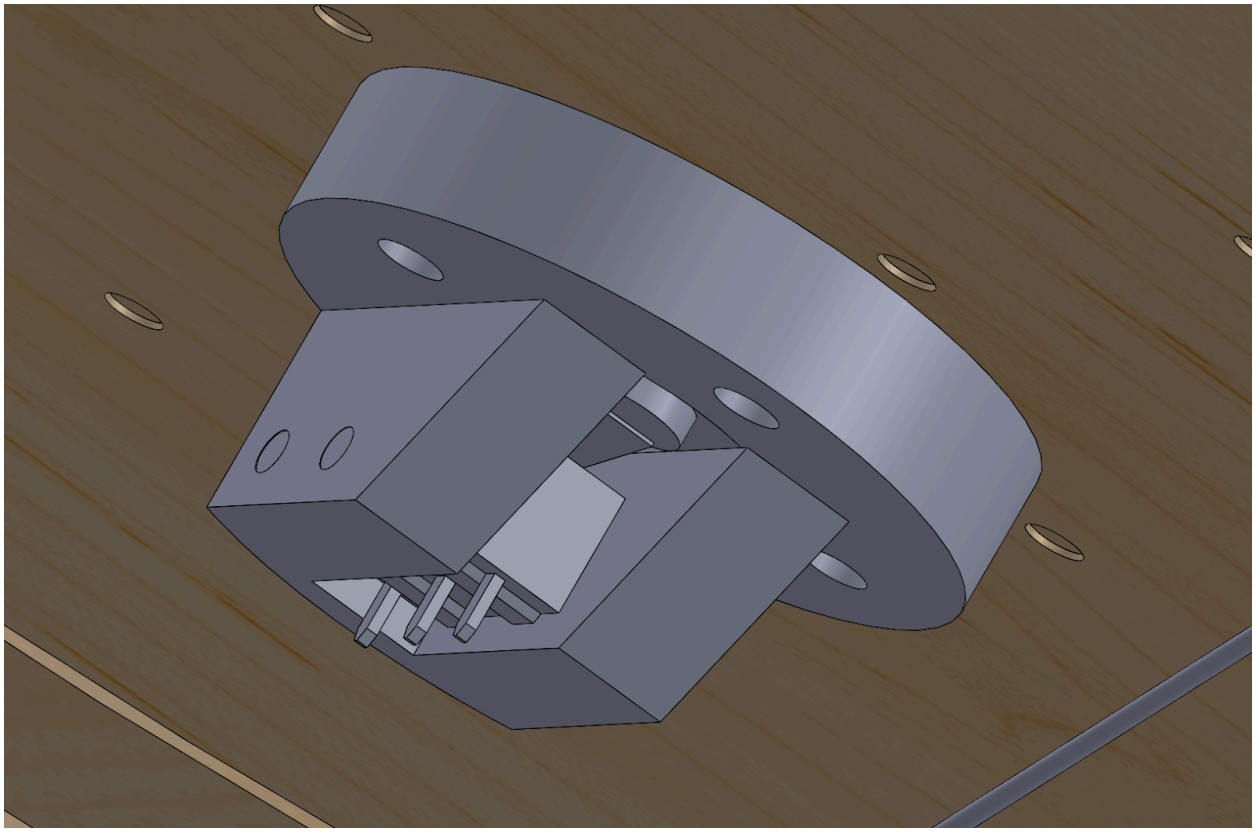
Detail: Transmission



Detail: Spur gears



Detail: String and shaft



Detail: Reset button. Note the limit switch in the middle

Appendix: Code

```
#include <Arduino.h>
#include <ESP32Encoder.h>

//Motor 1 parameters-----------------------------------------------------
//encoder1
ESP32Encoder encoder1;
#define ENCODER1_PIN1 27
#define ENCODER1_PIN2 33
//Motor pins
#define M1PWM 26
#define M1DIR 25
//Joystick pin
#define POT1 4


int thetaDes1 = 0;
```

```
int thetaMax1 = 3591.84 / 12;  // 74.83 * 42 counts per revolution divided
by 6 = 60° range
int theta01 = map(1887, 4095, 0, 0, thetaMax1);
int theta1 = theta01;   // initial position : flat maze
int D1 = 0;
int potReading1 = 0;
float Kp1 = 0.5;
float Ki1 = 0.0;
float kd1 = 0;
int IMax1 = 0;
float error1;
float prevError1 = 0;
float sum_error1;
float derivative1;
int potPrev1;
volatile int count1 = 0;  // encoder1 count

// setting PWM properties
const int freq1 = 20000;
const int resolution1 = 8;
const int MAX_PWM_VOLTAGE1 = 180;
const int MIN_PWM_VOLTAGE1 = 100;
const int D_MIN = 5;


//Motor 2 parameters-------------------------------------------------
//Encoder 2
ESP32Encoder encoder2;
#define ENCODER2_PIN1 14
#define ENCODER2_PIN2 32
//Motor pins
#define M2PWM 12
#define M2DIR 13
//Joystick pin
#define POT2 36


int thetaDes2 = 0;
int thetaMax2 = 3591.84 / 12;  // 74.83 * 42 counts per revolution = 60
deg range
```

```
int theta02 = map(1871, 4095, 0, 0, thetaMax1);
int theta2 = theta02;   // initial position : flat maze
int D2 = 0;
int potReading2 = 0;
float Kp2 = 0.5;
float Ki2 = 0.0;
float kd2 = 0;
int IMax2 = 0;
float error2;
float prevError2 = 0;
float sum_error2;
float derivative2;
int potPrev2;
volatile int count2 = 0;  // encoder2 count

// setting PWM properties
const int freq2 = 20000;
const int resolution2 = 8;
const int MAX_PWM_VOLTAGE2 = 180;
const int MIN_PWM_VOLTAGE2 = 100;


//Motor control Timer interrupt ----------------------------------------

volatile bool deltaT = false;  // check timer interrupt
hw_timer_t* timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

void IRAM_ATTR onTime() {
  portENTER_CRITICAL_ISR(&timerMux);
  count1 = encoder1.getCount();
  encoder1.clearCount();
  count2 = encoder2.getCount();
  encoder2.clearCount();
  deltaT = true;  // the function to be called when timer interrupt is
triggered
  portEXIT_CRITICAL_ISR(&timerMux);
}

void motorInit() {
```

```
  //Encoder setup
  ESP32Encoder::useInternalWeakPullResistors = puType::up;  // Enable the
weak pull up resistors
  encoder1.attachHalfQuad(ENCODER1_PIN1, ENCODER1_PIN2);    //27,33
// Attache pins for use as encoder pins
  encoder1.setCount(0);                                     // set
starting count value after attaching
  encoder2.attachHalfQuad(ENCODER2_PIN1, ENCODER2_PIN2);    //14,32
// Attache pins for use as encoder pins
  encoder2.setCount(0);                                     // set
starting count value after attaching

  // Timer
  timer = timerBegin(1000000);           // Set timer frequency to 1Mhz
  timerAttachInterrupt(timer, &onTime);  // Attach onTimer1 function to
our timer.
  timerAlarm(timer, 10000, true, 0);     // 10000 * 1 us = 10 ms,
autoreload true

  //Motor 1
  pinMode(POT1, INPUT);
  pinMode(M1DIR, OUTPUT);
  digitalWrite(M1DIR, LOW);
  ledcAttach(M1PWM, freq1, resolution1);  // configure PWM
functionalitites with attaching the channel to the GPIO to be controller

  //Motor 2
  pinMode(POT2, INPUT);
  pinMode(M2DIR, OUTPUT);
  digitalWrite(M2DIR, LOW);
  ledcAttach(M2PWM, freq2, resolution2);  // configure PWM
functionalitites with attaching the channel to the GPIO to be controller
}


//Other timer definition-----------------------------------------

//Define constants
#define JOYSTICKBTN 15  // declare the button ED pin number
#define WINBTN 34
```

```
//Setup variables
byte state = 0;  //Init

volatile bool JoystickbuttonIsPressed = false;
volatile bool joystickDEBOUNCINGflag = false;

volatile bool WinbuttonIsPressed = false;
volatile bool winDEBOUNCINGflag = false;

volatile bool timekeepingflag = false;


//Setup time variables
hw_timer_t* joystickdebouncetimer = NULL;  // Timer variable for joystick
button debouncing
portMUX_TYPE joystickdebouncetimerMux = portMUX_INITIALIZER_UNLOCKED;

hw_timer_t* windebouncetimer = NULL;  // Timer variable for winning button
debouncing
portMUX_TYPE windebouncetimerMux = portMUX_INITIALIZER_UNLOCKED;

hw_timer_t* gametimer = NULL;  // Timer variable for the game duration
portMUX_TYPE gametimerMux = portMUX_INITIALIZER_UNLOCKED;


//Joystick button
void IRAM_ATTR joystickonTime() {
  portENTER_CRITICAL_ISR(&joystickdebouncetimerMux);
  joystickDEBOUNCINGflag = false;  // the function to be called when timer
interrupt is triggered
  portEXIT_CRITICAL_ISR(&joystickdebouncetimerMux);
  timerStop(joystickdebouncetimer);
  JoystickbuttonIsPressed = false;
}

void IRAM_ATTR JoystickISR() {  // the function to be called when
interrupt is triggered
  JoystickbuttonIsPressed = true;
```

```
}

void JoystickTimerInterruptInit() {
  joystickdebouncetimer = timerBegin(1000000);                    // Set
timer frequency to 1Mhz
  timerAttachInterrupt(joystickdebouncetimer, &joystickonTime);  // Attach
onTimer function to our timer.
  // Set alarm to call onTime function every second (value in
microseconds).
  // Repeat the alarm (third parameter) with unlimited count = 0 (fourth
parameter).
  timerAlarm(joystickdebouncetimer, 300000, true, 0);
}


// Winning button
void IRAM_ATTR winonTime() {
  portENTER_CRITICAL_ISR(&windebouncetimerMux);
  winDEBOUNCINGflag = false;  // the function to be called when timer
interrupt is triggered
  portEXIT_CRITICAL_ISR(&windebouncetimerMux);
  timerStop(windebouncetimer);
  WinbuttonIsPressed = false;
}

void IRAM_ATTR WinISR() {  // the function to be called when interrupt is
triggered
  WinbuttonIsPressed = true;
}

void WinTimerInterruptInit() {
  windebouncetimer = timerBegin(1000000);              // Set timer
frequency to 1Mhz
  timerAttachInterrupt(windebouncetimer, &winonTime);  // Attach onTimer
function to our timer.
  // Set alarm to call onTime function every second (value in
microseconds).
  // Repeat the alarm (third parameter) with unlimited count = 0 (fourth
parameter).
  timerAlarm(windebouncetimer, 300000, true, 0);  //
```

```
}

//Game timer
void IRAM_ATTR GameonTime() {
  portENTER_CRITICAL_ISR(&gametimerMux);
  timekeepingflag = false;  // the function to be called when timer
interrupt is triggered
  portEXIT_CRITICAL_ISR(&gametimerMux);
  timerStop(gametimer);
}

void gameTimerInterruptInit() {
  gametimer = timerBegin(1000000);               // Set timer frequency to
1Mhz
  timerAttachInterrupt(gametimer, &GameonTime);  // Attach onTimer
function to our timer.
  // Set alarm to call onTime function every second (value in
microseconds).
  // Repeat the alarm (third parameter) with unlimited count = 0 (fourth
parameter).
  timerAlarm(gametimer, 45000000, true, 0);  // 60s timer
}

//Setup ----------------------------------------------------------
void setup() {
  Serial.begin(115200);


  pinMode(JOYSTICKBTN, INPUT_PULLUP);
  attachInterrupt(JOYSTICKBTN, JoystickISR, FALLING);
  JoystickTimerInterruptInit();

  pinMode(WINBTN, INPUT);
  attachInterrupt(WINBTN, WinISR, RISING);
  WinTimerInterruptInit();

  gameTimerInterruptInit();

  motorInit();
}
```

```
////////////////////////////////////////////////////////////////////////////
Main loop ----------------------------------
void loop() {

  plotControlData();

  switch (state) {

    case 0:  // Idle state

      if (CheckForJoystickButtonPress()) {
        timerRestart(gametimer);
        JoystickButtonResponse();
        StartGameTimer();
        state = 1;
      }
      break;

    case 1:  // Playing state
       Playing();  // Control of the motors with the joystick

      if (CheckForWinningButtonPress()) {
        WinButtonResponse();
        state = 0;
      }
      if (CheckTimeIsOver()) {
        TimeOverResponse();
        state = 0;
      }

      break;

    default:  // ERROR
      Serial.println("SM_ERROR");
      break;
  }
}
////////////////////////////////////////////////////////////////////////////
```

```
// Function librairy --------------------------


//Joystick button
bool CheckForJoystickButtonPress() {
  if (JoystickbuttonIsPressed && joystickDEBOUNCINGflag == false) {

    portENTER_CRITICAL(&joystickdebouncetimerMux);
    joystickDEBOUNCINGflag = true;
    portEXIT_CRITICAL(&joystickdebouncetimerMux);
    timerStart(joystickdebouncetimer);

    return true;

  } else {
    return false;
  }
}


void JoystickButtonResponse() {
  WinbuttonIsPressed = false;  // If winning button was pressed during
state 0, prevent from winning without playing
  JoystickbuttonIsPressed = false;
  Serial.println("Game on !");
  delay(1000);  //Just for demo
}

//Winning button
bool CheckForWinningButtonPress() {
  if (WinbuttonIsPressed && winDEBOUNCINGflag == false) {
    portENTER_CRITICAL(&windebouncetimerMux);
    winDEBOUNCINGflag = true;
    portEXIT_CRITICAL(&windebouncetimerMux);
    timerStart(windebouncetimer);
    return true;
  } else {
    return false;
  }
}
```

```
void WinButtonResponse() {
  WinbuttonIsPressed = false;
  Serial.println("You won !");
  StopMotors();
  delay(5000);  //Just for demo
}

//Time keeping
void StartGameTimer() {
  portENTER_CRITICAL(&gametimerMux);
  timekeepingflag = true;
  portEXIT_CRITICAL(&gametimerMux);
  timerStart(gametimer);
}

bool CheckTimeIsOver() {
  if (timekeepingflag) {
    return false;
  } else {
    return true;
  }
}

void TimeOverResponse() {
  Serial.println("Time is up, game over!");
  timekeepingflag = false;
  StopMotors();
  delay(5000);  //Just for demo
}


// Motor control
void StopMotors() {
  ledcWrite(M1PWM, 0);
  ledcWrite(M2PWM, 0);
}
```

```
void Playing() {
  if (deltaT) {
    portENTER_CRITICAL(&timerMux);
    deltaT = false;
    portEXIT_CRITICAL(&timerMux);

    // control Motor 1-----------------------------------------------------
    // Read motor 1 postion
    theta1 += count1;

    // Read position 1 from joystick
    potReading1 = analogRead(POT1);

    //Debouncing joystick input
    if (abs(potReading1 - potPrev1) < 200) potReading1 = potPrev1;
    potPrev1 = potReading1;

    // Compute desired position
    thetaDes1 = map(potReading1, 4095, 0, 0, thetaMax1);

    // Compute error
    error1 = thetaDes1 - theta1;
    sum_error1 += error1;
    derivative1 = error1 - prevError1;
    prevError1 = error1;

    float max_sum_error1 = 10.0;
    if (sum_error1 > max_sum_error1) sum_error1 = max_sum_error1;
    if (sum_error1 < -max_sum_error1) sum_error1 = -max_sum_error1;

    // compute PWM Duty
    D1 = Kp1 * error1;//+ Ki1 * sum_error1 + kd1 * derivative1;

    //Ensure that D1 stays between maximum and minimum
    if (D1 > MAX_PWM_VOLTAGE1) {
      D1 = MAX_PWM_VOLTAGE1;
    } else if (D1 < -MAX_PWM_VOLTAGE1) {
      D1 = -MAX_PWM_VOLTAGE1;
    }
    if (D1 >= D_MIN) {
```

```
  if (D1 > 0 && D1 < MIN_PWM_VOLTAGE1) {
    D1 = MIN_PWM_VOLTAGE1;
  } else if (D1 < D_MIN) {
    D1 = 0;
  }
} else {
  if (abs(D1) > D_MIN && abs(D1) < MIN_PWM_VOLTAGE1) {
    D1 = -MIN_PWM_VOLTAGE1;
  } else if (abs(D1) < D_MIN) {
    D1 = 0;
  }
}

// Inject PWM value to motor 1
if (D1 > 0) {
  ledcWrite(M1PWM, D1);
  digitalWrite(M1DIR, 1);
} else if (D1 < 0) {
  ledcWrite(M1PWM, abs(D1));
  digitalWrite(M1DIR, 0);
} else {
  ledcWrite(M1PWM, 0);
  digitalWrite(M1DIR, 0);
}

// control Motor 2-----------------------------------------------------

// Read motor 2 postion
theta2 += count2;

// Read position 2 from joystick
potReading2 = analogRead(POT2);

//Debouncing joystick input
if (abs(potReading2 - potPrev2) < 200) potReading2 = potPrev2;
potPrev2 = potReading2;

// Compute desired position
thetaDes2 = map(potReading2, 4095,0 , 0, thetaMax2);
```

```
// Compute error
error2 = thetaDes2 - theta2;
sum_error2 += error2;
derivative2 = error2 - prevError2;
prevError2 = error2;

float max_sum_error2 = 10.0;
if (sum_error2 > max_sum_error2) sum_error2 = max_sum_error2;
if (sum_error2 < -max_sum_error2) sum_error2 = -max_sum_error2;

// Compute PWM value for motor 2
D2 = Kp2 * error2; //+ Ki2 * sum_error2 + kd2 * derivative2;

//Ensure that D2 stays between maximum and minimum
if (D2 > MAX_PWM_VOLTAGE2) {
  D2 = MAX_PWM_VOLTAGE2;
} else if (D2 < -MAX_PWM_VOLTAGE2) {
  D2 = -MAX_PWM_VOLTAGE2;
}
if (D2 >= D_MIN) {
  if (D2 > 0 && D2 < MIN_PWM_VOLTAGE2) {
    D2 = MIN_PWM_VOLTAGE2;
  } else if (D2 < D_MIN) {
    D2 = 0;  // Added semicolon
  }
} else {
  if (abs(D2) > D_MIN && abs(D2) < MIN_PWM_VOLTAGE2) {
    D2 = -MIN_PWM_VOLTAGE2;
  } else if (abs(D2) < D_MIN) {
    D2 = 0;  // Added semicolon
  }
}

// Inject PWM value to motor 2
if (D2 > 0) {
  ledcWrite(M2PWM, D2);
  digitalWrite(M2DIR, 1);
} else if (D2 < 0) {
  ledcWrite(M2PWM, abs(D2));
  digitalWrite(M2DIR, 0);
```

```
    } else {
      ledcWrite(M2PWM, 0);
      digitalWrite(M2DIR, 0);
    }
  }

  plotControlData();
}


void plotControlData() {

  Serial.print("Machine state : ");
  Serial.print(state);
  Serial.print("     ");

  Serial.print("Position 1:");
  Serial.print(theta1);
  Serial.print(" ");
  Serial.print("Desired_Position 1:");
  Serial.print(thetaDes1);
  Serial.print(" ");
  Serial.print("PWM_Duty 1:");
  Serial.print(D1);
  Serial.print("     ");

  Serial.print("Position 2:");
  Serial.print(theta2);
  Serial.print(" ");
  Serial.print("Desired_Position 2:");
  Serial.print(thetaDes2);
  Serial.print(" ");
  Serial.print("PWM_Duty 2:");
  Serial.println(D2);
}
```

MEC ENG 102B

Mechatronics Design

Fall 2024 – Project

Final deliverables

*Create both a **single PDF file** and a **single .MP4 or .MOV file** to upload to bcourses by the deadline.*
Only one person submits these files; include all teammate names.
It's been a pleasure to see your growth, dedication and creativity this semester. These last project deliverables are your opportunity to show off your great accomplishments and skill online.
It also represents the assessment of the final 7% of your course grade.

– Showcase (0.5% of overall grade) –

The Showcase for Mechatronics Design is on **12/11 from 4-5:30pm in Jacobs 310**. You should arrive early, and are open to set up at 3:45p, and you will have until 5:45p to clean up. Participation/attendance for the full duration of the showcase is required. This is a wonderful opportunity to see and share the great creativity and hard work you've performed this semester.

Also, remember there is a project award competition during the showcase this semester. Sign up for a category using this link (**by 12/10 at 11:59a -- NOON**). All teams strongly encouraged to participate!

– Report (5% of overall grade, due 12/19 at 10:00p) –

Compile a "Manual" for your fully integrated device, such that another engineer could replicate, and even iterate upon, your project. The main body of the document (not including the appendix) should be *<3 pages at 11pt font, including images* – prioritize and condense your images and descriptions to keep it to-the-point, and focused on the elements we are assessing you on. This report will be posted online at the project archive, along with your video (instruction below). Within 3-pages you must include:

- An updated version of the "Opportunity" that this device addresses (from P2).
- An updated version of your device's high-level strategy (from P2) to match your realized product. In addition, discuss/compare your initial desired functionality to your achieved specifications (e.g., we wanted to drive at least 1 foot per second and achieved a maximum speed of 0.95 feet per second).
- A photo showing the integrated physical device, fully assembled and with labels showing where actuators and sensors are located and highlighting any moving parts/joints. You can include a couple views to show key subsystems as needed. Think of this diagram as a "key" of terms that you can use in your description (e.g., "the wheel transmission..." highlight in the image what this means to you).
- Discuss the function-critical decisions you made (only as related to the required project elements). Report your function-critical calculations (e.g. forces on bearings, required motor torque/speed, etc.) and relate these to specifications of the specific load-critical mechanical and actuation components that meet those requirements. *Calculations should be typed up, not handwritten*. You will be evaluated on both the relevance and completeness of the calculations performed, along with the appropriateness of the selected components to meet these needs.
- Your updated circuit diagram (updated from P5) and State Transition Diagram (updated from P5) that matches with the program that you wrote for your machine.
- A short, couple sentence, reflection for future students in the class. Discuss both the strategies that worked well for your group, as well as "what we wish we had done differently" debrief of your experience on this project to assist future students.

After the initial 3 pages, you must then include these Appendices (graded, with no page limit):

1MEC ENG 102B

Mechatronics Design

Fall 2024 – Project

Final deliverables

○ Complete Bill of Materials for the final product only (updated from P4)
○ Images of the CAD, showing mechanical transmission elements (updated from P4)
○ Screen shots of your entire code, showing your event-driven program (updated from P5).

You can include any additional appendices that you choose, e.g., to discuss functions that are beyond the project requirements (e.g. camera or battery integration), however they will not necessarily be reviewed or assessed by the teaching team.

– Video (1% of overall grade, due 12/19 at 10:00p) –

You will produce a short video (**no shorter than 1:30 and no longer than 3:00**) highlighting the

functionality of the complete, fully assembled project. It will be linked on the project archive. Include the following:
(1) briefly describing the motivation for the device,
(2) describing/showing how it typically works,
Do not use copyrighted materials, which would prevent them from getting posted.
The video must be uploaded as an .mp4 or .mov file (not a website or cloud link!) and be ***< 50MB.***
Hint: you can compress videos by uploading and downloading them from YouTube.
– Peer- and self-evaluation (0.5% of overall grade, due 12/19 at 11:59p) –
All students must provide a self and peer evaluation. The survey, posted later via bcourses, will contain questions regarding both your own contributions as well as your teammates' contributions. This can factor into your final grade in the class, primarily in extreme cases.
Also, remember to:
- complete course evaluations for potential extra credit, and
- submit all ASME Maker Grant requirements on time to be eligible for reimbursement.