Automatic Card Dealer

Group 6: Arjun Chauhan, Vinh Dinh, Samuel Harris, and Derek Rodriguez

Opportunity

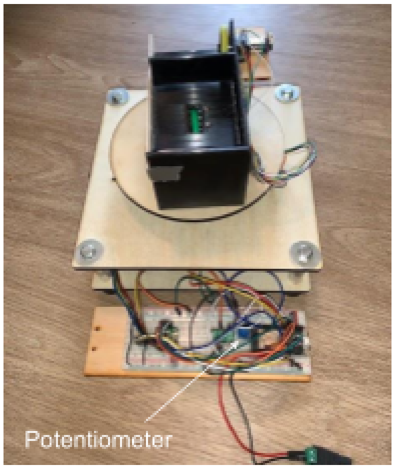
The essence of card games rests not just in the strategic movements but also in the shared moments of delight and camaraderie they foster. These games are more than just enjoyable activities for people of all abilities; they are bridges that connect hearts, brains, and traditions. However, the accessibility of such encounters is frequently hampered. Enter the automatic card dealer, which is more than simply a gizmo; it is a portal to inclusivity. It breaks down boundaries, allowing everyone, regardless of physical skill, to participate in the deal of a deck. Its modest mechanism has the potential to amp up the laughter, competition, and connection that card games provide, embracing a broad tapestry of abilities and building a truly inclusive world of play.

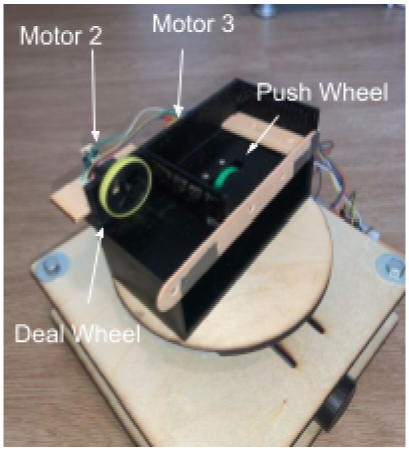
High-Level Strategy

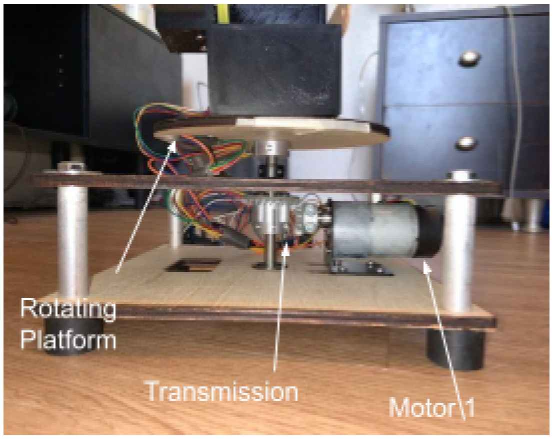
We designed our card dealing mechanism to sit on a rotating plate which could be adjusted by potentiometer to the appropriate number of players. Once the deck is loaded, the button is pushed and the system is initiated. Cycling through on a timer, at each player, the plate pauses, the first wheel on the card mechanism slowly pulls the bottom card forward, as it passes under the barrier that stops multiple cards from existing simultaneously, the second wheel quickly spins, grabbing and shooting the card to the player. Although this second wheel continues to rotate throughout the cycle, by now the first has been stopped, and the plate rotates to the next player where this process is repeated. At the last player, the plate rotates back to its original position and by pushing the button the cycle begins again.

Our final design was very different from where we began in P2. Initially, we were overambitious, wanting to integrate a card shuffler and dealer which was just not feasible in one semester. The card mechanism changed from initially one-wheel pulling and shooting to the two-wheel design we ended with. We also anticipated the need for an exit chute which did not end up being necessary.

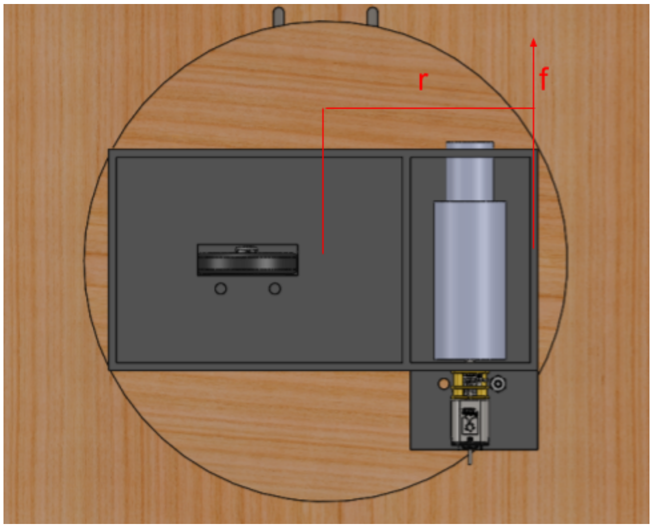
Integrated Physical Design







Function-Critical Decisions



For our project, having DC motors that can produce sufficient torque is highly important. To rotate the platform, the large motor is borrowed from Hesse shop with the expectation of creating the significantly higher torque that is required. The base motor is able to safely produce a maximum torque of 79 kg-mm, and it is paired with a 1:4 gear transmission. As a result, the final output torque produced is t_f = 316 kg-mm. The rotating platform is estimated with the length side 100 mm implying the distance from center r = 50 mm. Using the motor, the force at 50 mm from the platform center is

$$f = \frac{t_f}{r} = \frac{316}{50} = 6.32\, kg \sim 63.2\, N$$

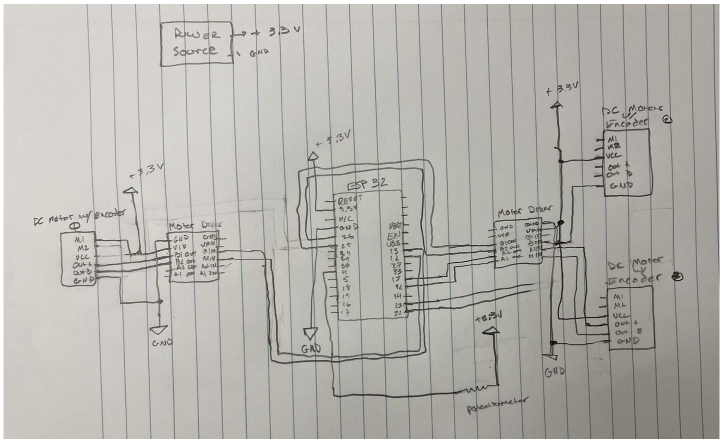
With the mass estimation at the edge of the top rotating platform to be m = 0.5 kg, so the linear acceleration at maximum:

$$a = \frac{f}{m} = 126.4\, m/s^2$$

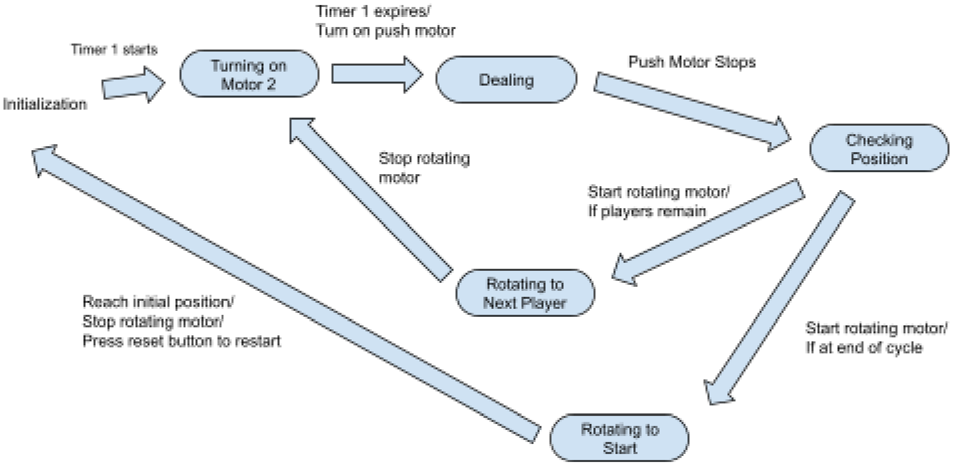
The maximum acceleration is not necessarily to be reached or used, but it states the capability of the motor for our project.

The dealing mechanism has two micro DC motors from the micro kit. To deal one card at a time, these motors are able to produce sufficient torque as one card is estimated to weigh only 2 g, which is highly light.

Circuit Diagram



State Transition Diagram



Final Thoughts

At the conclusion of this project, our team can take away a handful of lessons including the unpredictability of integrating different parts of our project, the variability in seemingly identical trials, the complexity of running multiple motors and timers simultaneously, and many more. But if there were to be one key takeaway here, it would undoubtedly be time management. Deliverables P1-P5 resulted in a consistent workflow that broke down a seemingly insurmountable task into bite-sized chunks. However, once P5 is submitted, you are in no way finished with your mechanism. This was learned somewhere between hours 16 and 18 on the Tuesday before final presentations in Hesse lobby. We spent seemingly equal amounts of time and brain power in the final 60 hours before our functionality test as in the 12 weeks leading up to it. I think we can all agree that although we were able to get the results we wanted, we could have done so in a far more efficient way. You can not overestimate the amount of time you will need to spend on adjustments, bugs, and problems that never crossed your mind until they are the only thing between you and a functioning mechanism. So, plan ahead, and give yourself leeway! Oh and go to lecture, we promise it will make things much much easier.

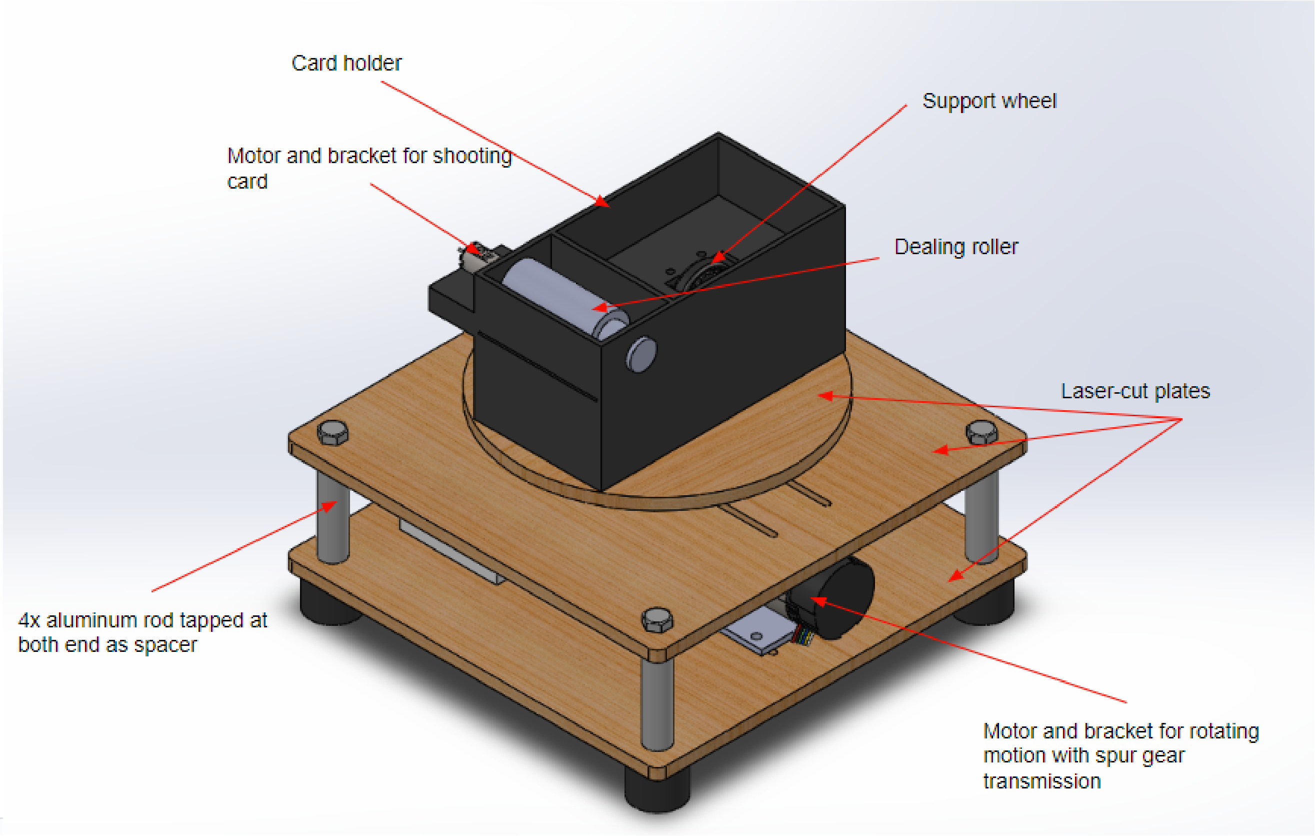
APPENDICES

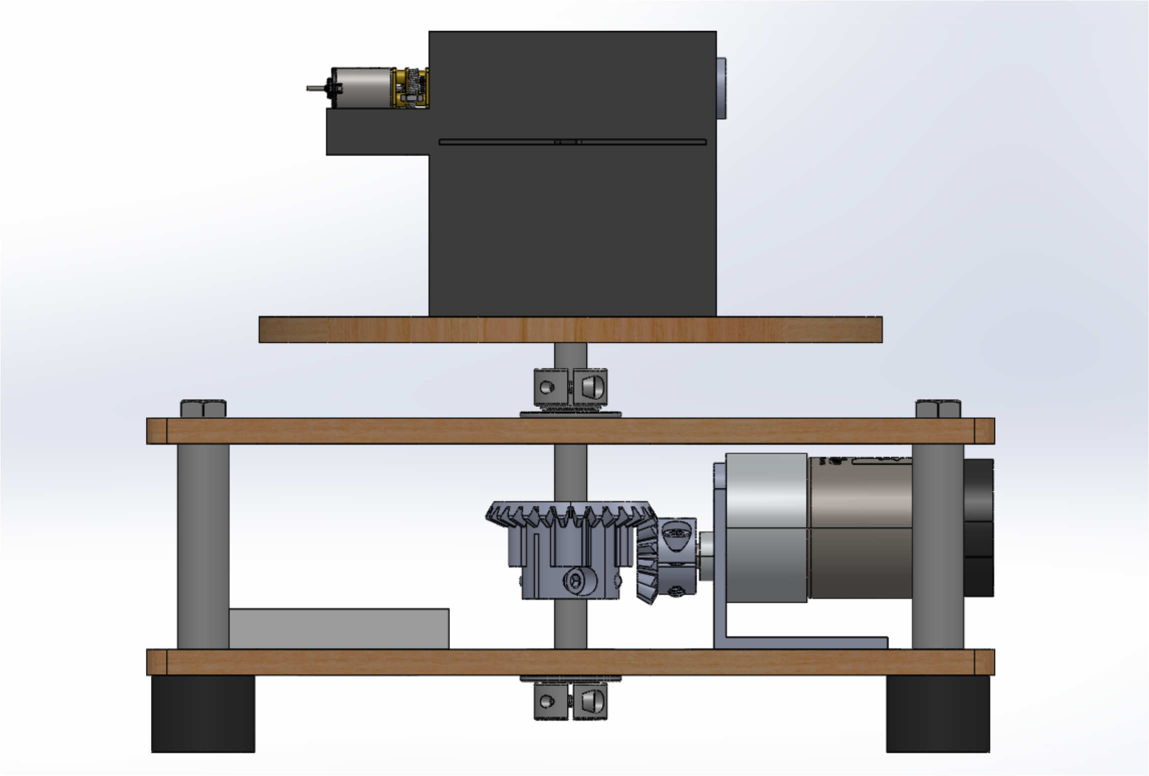
Appendix A: Bill of Materials

Location	Item	Amount	Price($)	Vendor	Link
Base	DC Geared Motor with Encoder	1	Hesse shop	DFRobot	https://www.dfrobot.com/product-634.html
Base	Bevel Gear Set	1	29.99	ServoCity	https://www.servocity.com/2-1-ratio-bevel-gear-set-6mm-d-bore-pinion-gear/
Base	Hyper Hub	1	7.99	ServoCity	https://www.servocity.com/1310-series-hyper-hub-8mm-bore/
Base	Round Shaft	1	4.49	ServoCity	https://www.servocity.com/8mm-x-250mm-stainless-steel-precision-shafting/
Base	Motor Mounting Bracket	1	8.99	Amazon	https://www.amazon.com/gp/product/B00TK0X03U/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1
Base	1/4" thick Plywood Sheet	18” x 30”	6.25	Jacobs Store	https://store.jacobshall.org/products/plywood-1-4-x-18-x-30
Base	Threaded Standoff	4	3.44 x 4	McMaster	https://www.mcmaster.com/93330A672/
Base	¼ - 20 Bolt	4	Owned	N/A	N/A
Base	¼ - 20 Washer	4	Owned	N/A	N/A
Base	Threaded Bumper	4	3.14 x 4	McMaster	https://www.mcmaster.com/93115K121/
Base	Flanged Ball Bearing	2	9.20 x 2	McMaster	https://www.mcmaster.com/57155K513/
Base	Clamping Shaft Collar	2	9.35 x 2	McMaster	https://www.mcmaster.com/6063K14/
Base	Round Shim	1 pack of 25	9.12	McMaster	https://www.mcmaster.com/98089A336/
Base	Disc Spring	1 pack of 12	4.11	McMaster	https://www.mcmaster.com/96445K35/

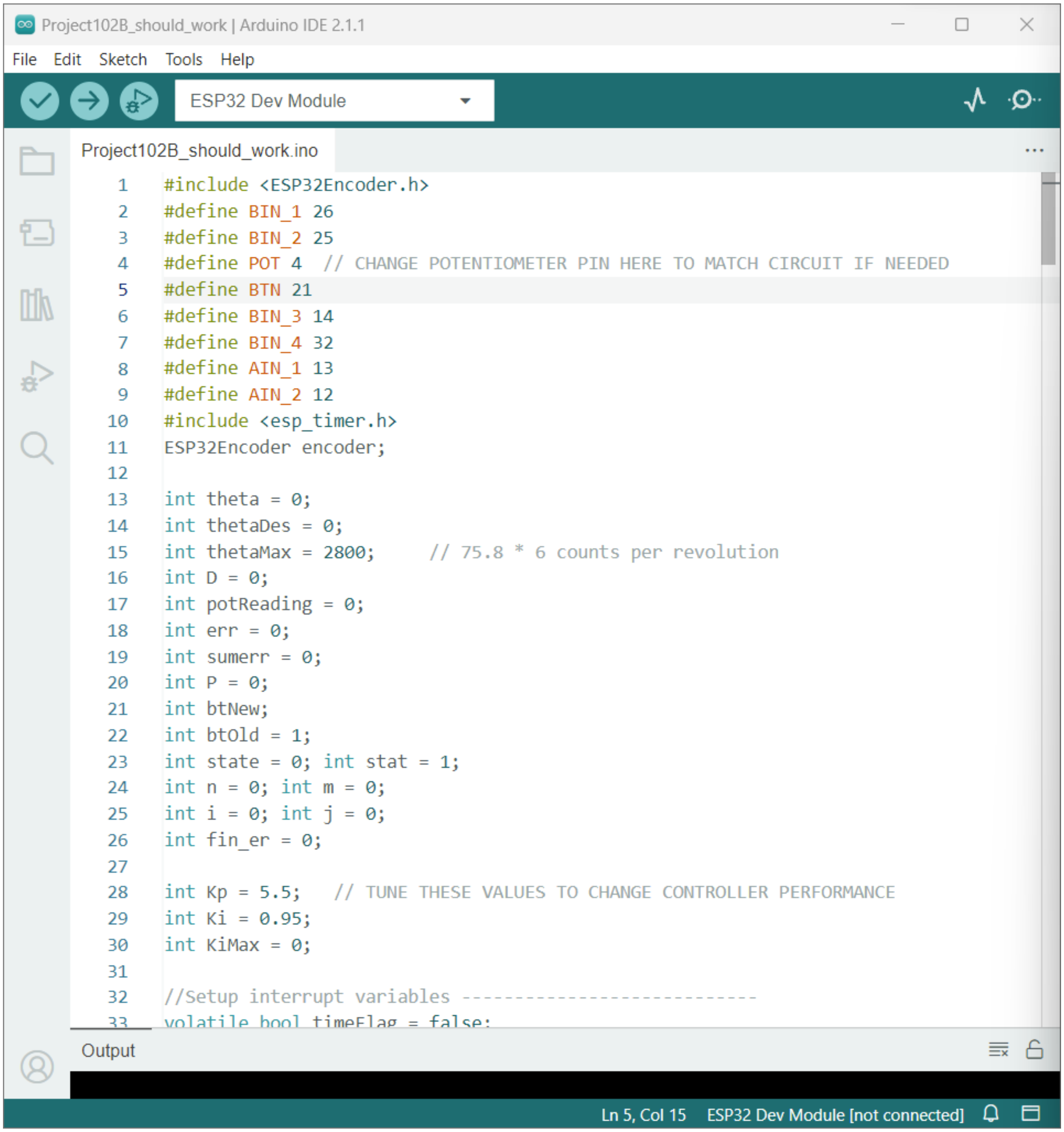
Card Mechanism	Machine Hex Nut: #2-56	1 pack of 25	2.25	Pololu	https://www.pololu.com/product/1067/specs
Card Mechanism	Machine Screw: #2-56, 7/16″	1 pack of 25	1.39	Pololu	https://www.pololu.com/product/1067/specs
Card Mechanism	Micro Gearmotor HPCB 12V with Extended Shaft	2	Owned	Pololu	https://www.pololu.com/product/3053
Card Mechanism	Pololu Wheel 32×7mm	pack of 2	3.95	Pololu	https://www.pololu.com/product/1087

Appendix B: CAD

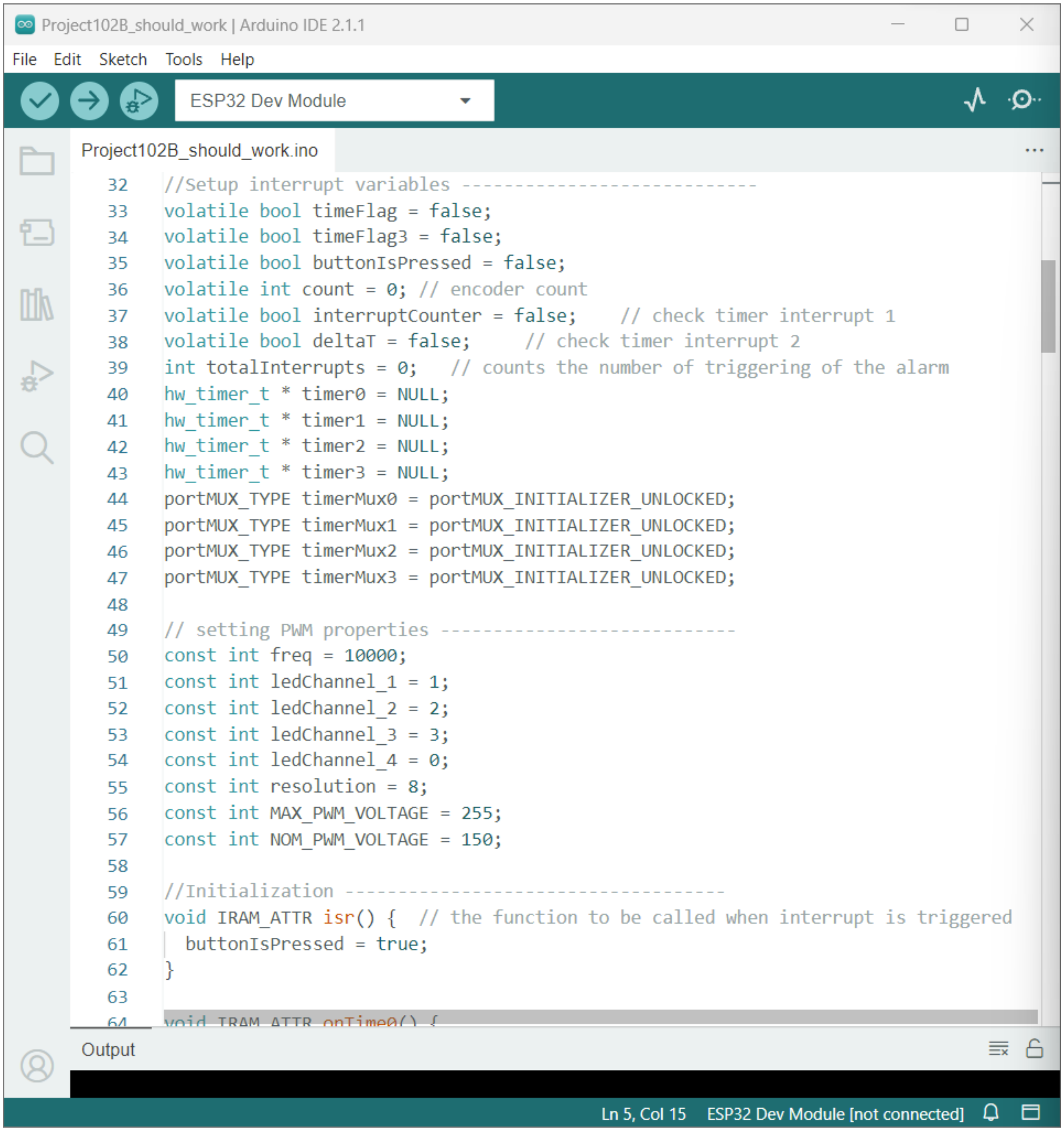

Card holder
Support wheel
Motor and bracket for shooting card
Dealing roller
Laser-cut plates
4x aluminum rod tapped at both end as spacer
Motor and bracket for rotating motion with spur gear transmission



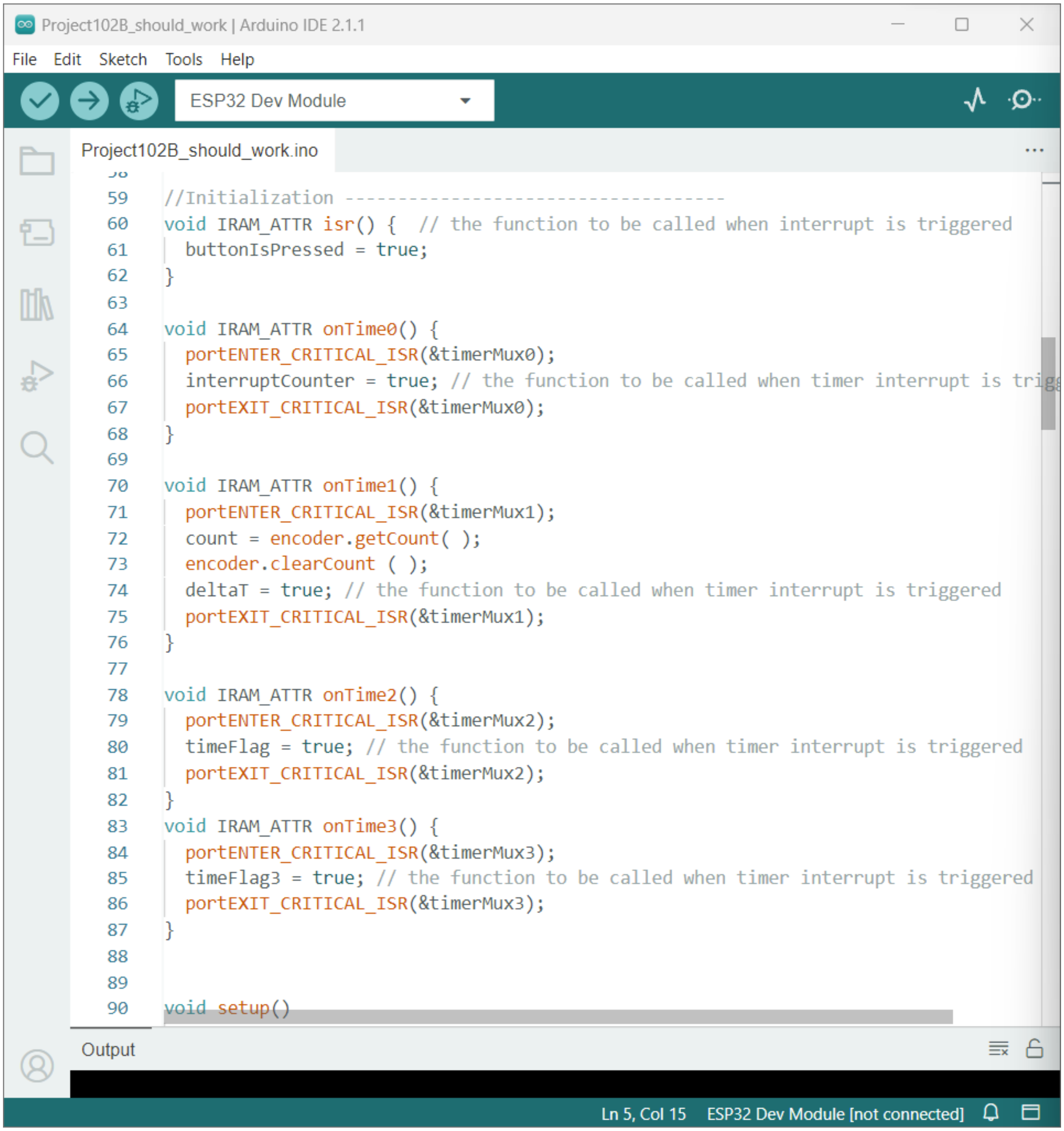
Appendix C: Full Code


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15
ESP32 Dev Module [not connected]

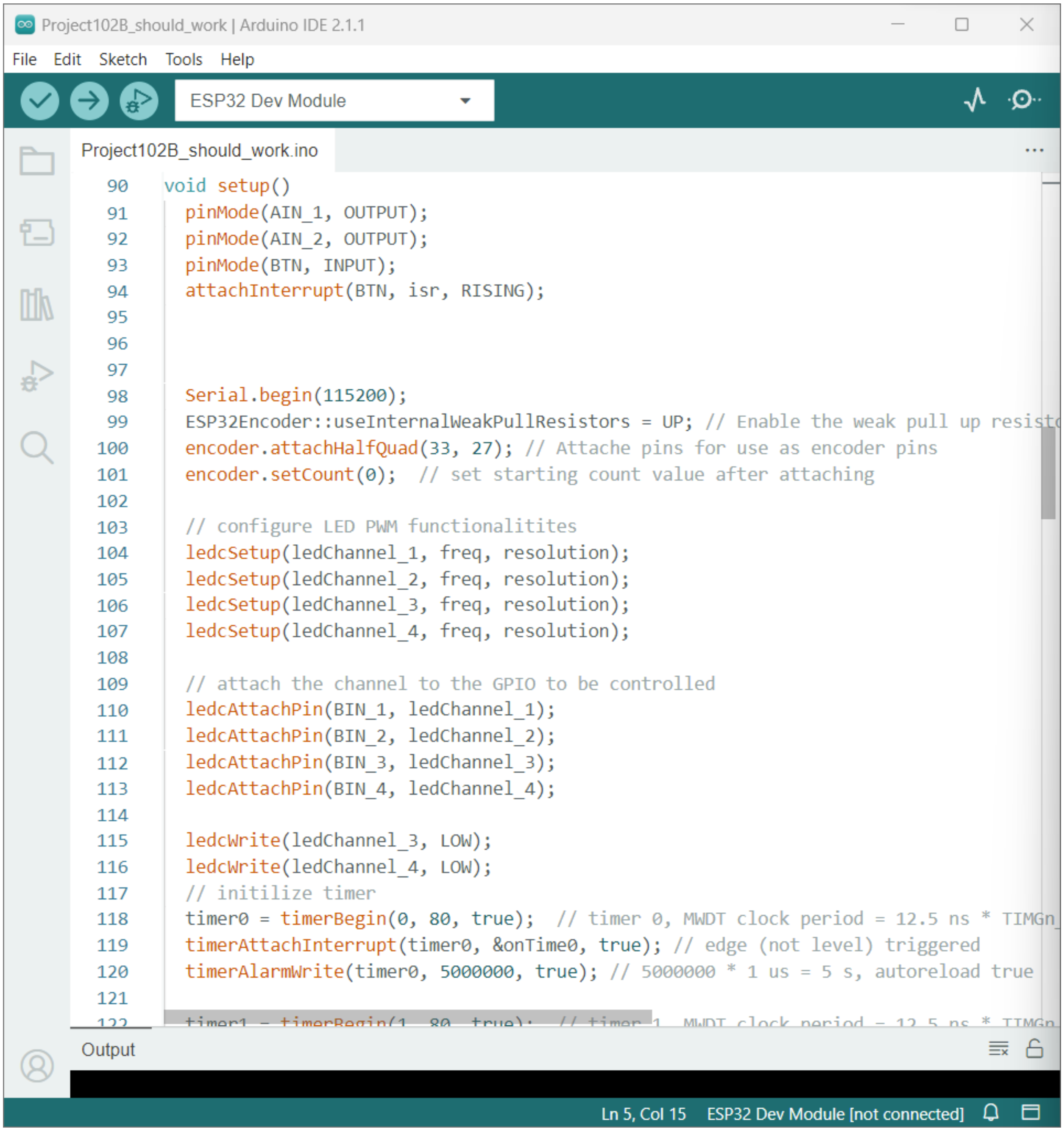
```
#include <ESP32Encoder.h>
#define BIN_1 26
#define BIN_2 25
#define POT 4  // CHANGE POTENTIOMETER PIN HERE TO MATCH CIRCUIT IF NEEDED
#define BTN 21
#define BIN_3 14
#define BIN_4 32
#define AIN_1 13
#define AIN_2 12
#include <esp_timer.h>
ESP32Encoder encoder;

int theta = 0;
int thetaDes = 0;
int thetaMax = 2800;     // 75.8 * 6 counts per revolution
int D = 0;
int potReading = 0;
int err = 0;
int sumerr = 0;
int P = 0;
int btNew;
int btOld = 1;
int state = 0; int stat = 1;
int n = 0; int m = 0;
int i = 0; int j = 0;
int fin_er = 0;

int Kp = 5.5;   // TUNE THESE VALUES TO CHANGE CONTROLLER PERFORMANCE
int Ki = 0.95;
int KiMax = 0;

//Setup interrupt variables ----------------------------
volatile bool timeFlag = false;
```


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15
ESP32 Dev Module [not connected]

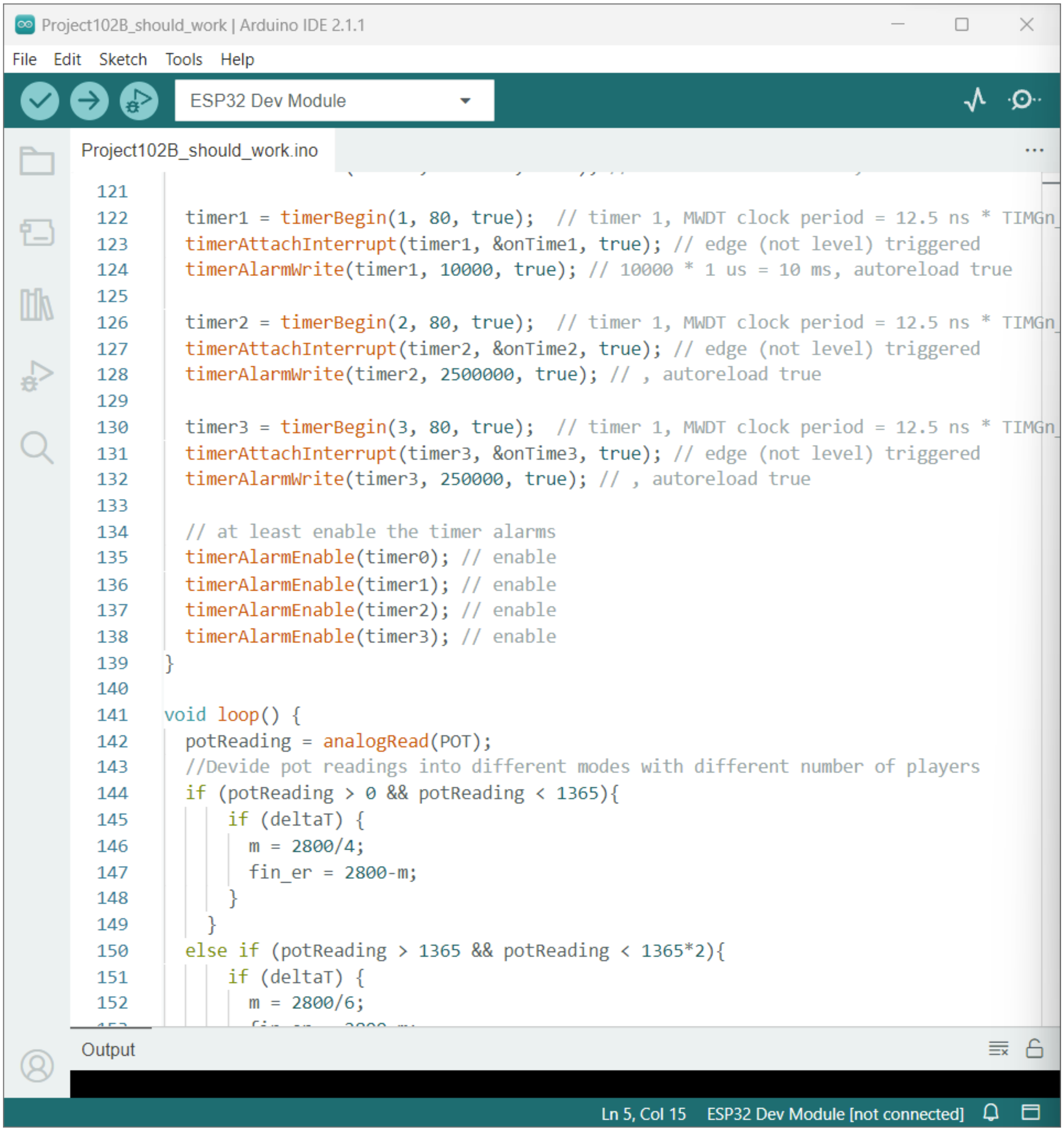
```
//Setup interrupt variables ----------------------------
volatile bool timeFlag = false;
volatile bool timeFlag3 = false;
volatile bool buttonIsPressed = false;
volatile int count = 0; // encoder count
volatile bool interruptCounter = false;    // check timer interrupt 1
volatile bool deltaT = false;     // check timer interrupt 2
int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer0 = NULL;
hw_timer_t * timer1 = NULL;
hw_timer_t * timer2 = NULL;
hw_timer_t * timer3 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux2 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux3 = portMUX_INITIALIZER_UNLOCKED;

// setting PWM properties ----------------------------
const int freq = 10000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int ledChannel_3 = 3;
const int ledChannel_4 = 0;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;

//Initialization ------------------------------------
void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  buttonIsPressed = true;
}

void IRAM_ATTR onTime0() {
```


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15
ESP32 Dev Module [not connected]

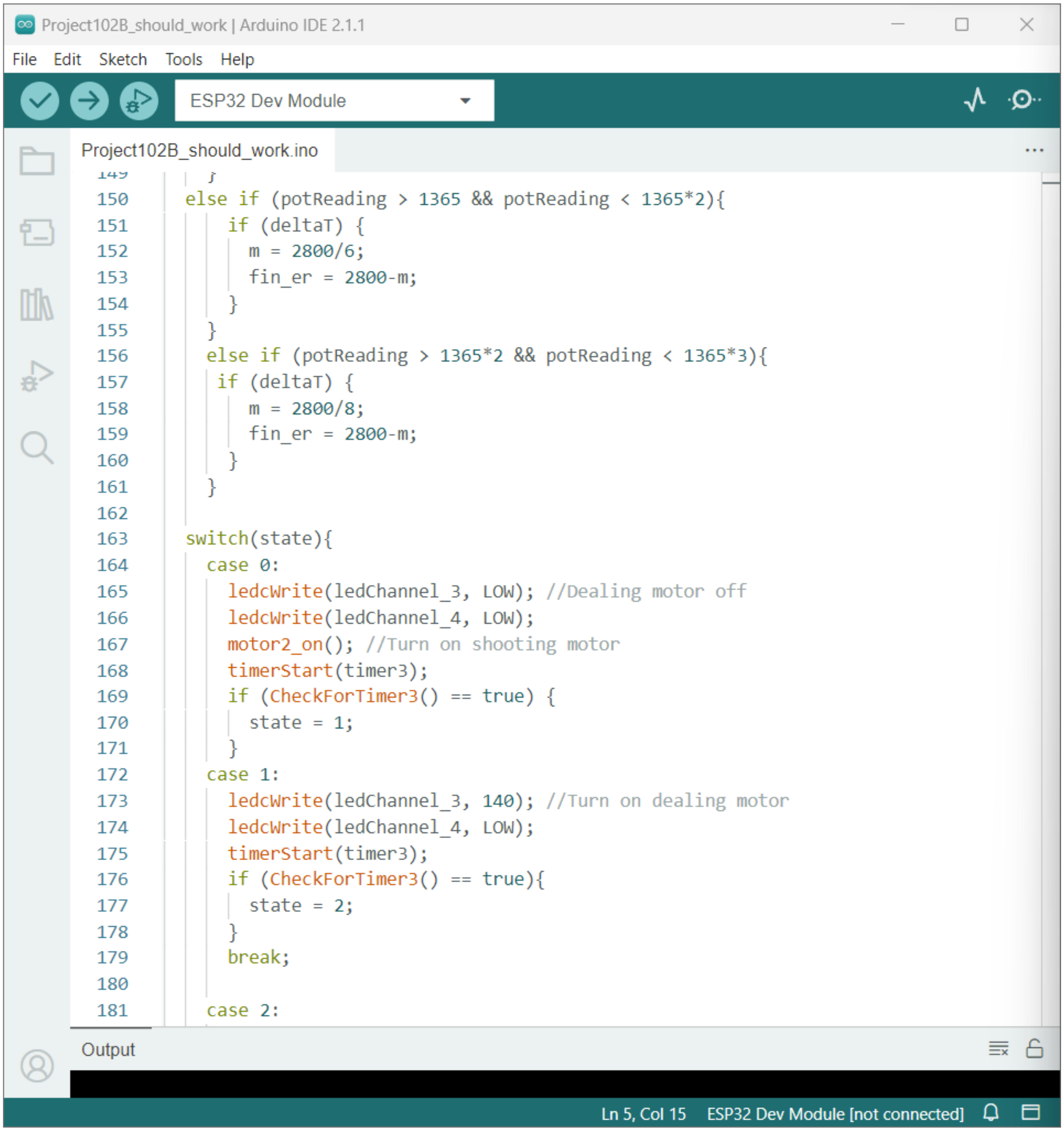
```
//Initialization -----------------------------------
void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  buttonIsPressed = true;
}

void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  interruptCounter = true; // the function to be called when timer interrupt is trig
  portEXIT_CRITICAL_ISR(&timerMux0);
}

void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  count = encoder.getCount( );
  encoder.clearCount ( );
  deltaT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}

void IRAM_ATTR onTime2() {
  portENTER_CRITICAL_ISR(&timerMux2);
  timeFlag = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux2);
}
void IRAM_ATTR onTime3() {
  portENTER_CRITICAL_ISR(&timerMux3);
  timeFlag3 = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux3);
}


void setup()
```


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15
ESP32 Dev Module [not connected]

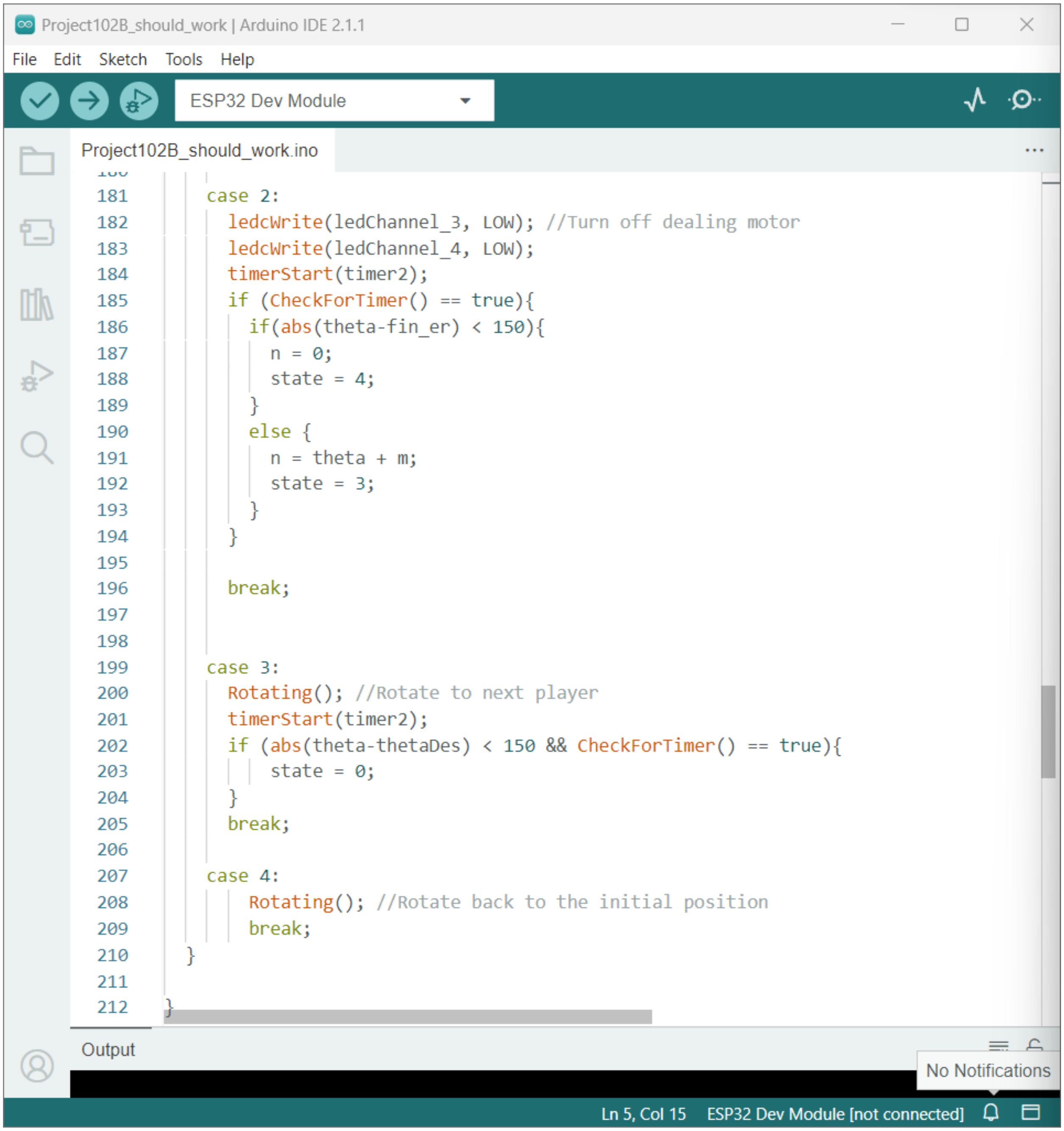
```
void setup()
  pinMode(AIN_1, OUTPUT);
  pinMode(AIN_2, OUTPUT);
  pinMode(BTN, INPUT);
  attachInterrupt(BTN, isr, RISING);



  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resisto
  encoder.attachHalfQuad(33, 27); // Attache pins for use as encoder pins
  encoder.setCount(0);  // set starting count value after attaching

  // configure LED PWM functionalitites
  ledcSetup(ledChannel_1, freq, resolution);
  ledcSetup(ledChannel_2, freq, resolution);
  ledcSetup(ledChannel_3, freq, resolution);
  ledcSetup(ledChannel_4, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);
  ledcAttachPin(BIN_3, ledChannel_3);
  ledcAttachPin(BIN_4, ledChannel_4);

  ledcWrite(ledChannel_3, LOW);
  ledcWrite(ledChannel_4, LOW);
  // initilize timer
  timer0 = timerBegin(0, 80, true);  // timer 0, MWDT clock period = 12.5 ns * TIMGn_
  timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
  timerAlarmWrite(timer0, 5000000, true); // 5000000 * 1 us = 5 s, autoreload true

  timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn
```


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15
ESP32 Dev Module [not connected]

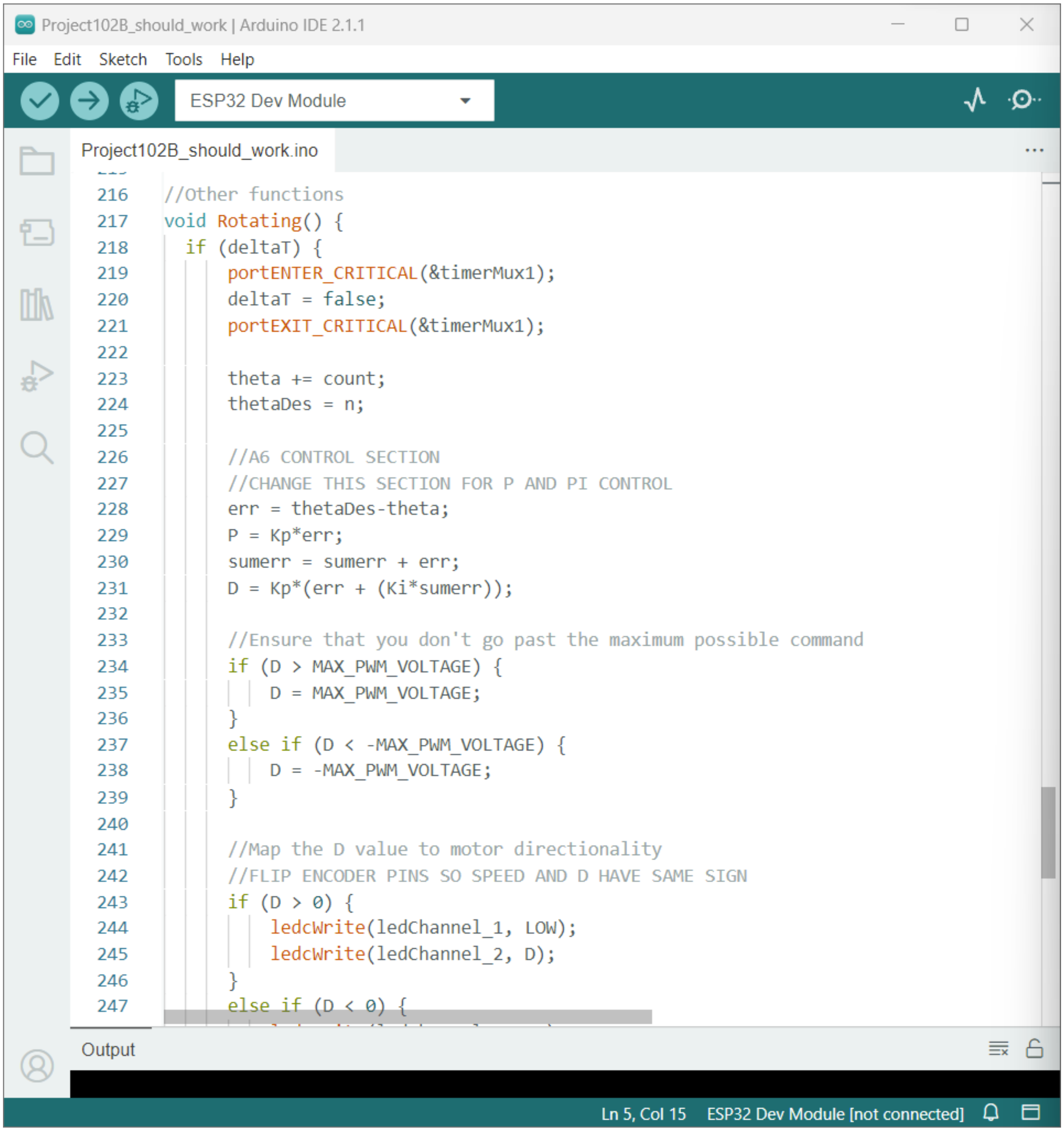
```

  timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 10000, true); // 10000 * 1 us = 10 ms, autoreload true

  timer2 = timerBegin(2, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_
  timerAttachInterrupt(timer2, &onTime2, true); // edge (not level) triggered
  timerAlarmWrite(timer2, 2500000, true); // , autoreload true

  timer3 = timerBegin(3, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_
  timerAttachInterrupt(timer3, &onTime3, true); // edge (not level) triggered
  timerAlarmWrite(timer3, 250000, true); // , autoreload true

  // at least enable the timer alarms
  timerAlarmEnable(timer0); // enable
  timerAlarmEnable(timer1); // enable
  timerAlarmEnable(timer2); // enable
  timerAlarmEnable(timer3); // enable
}

void loop() {
  potReading = analogRead(POT);
  //Devide pot readings into different modes with different number of players
  if (potReading > 0 && potReading < 1365){
    if (deltaT) {
      m = 2800/4;
      fin_er = 2800-m;
    }
  }
  else if (potReading > 1365 && potReading < 1365*2){
    if (deltaT) {
      m = 2800/6;
```


Project102B_should_work | Arduino IDE 2.1.1
File Edit Sketch Tools Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15 ESP32 Dev Module [not connected]

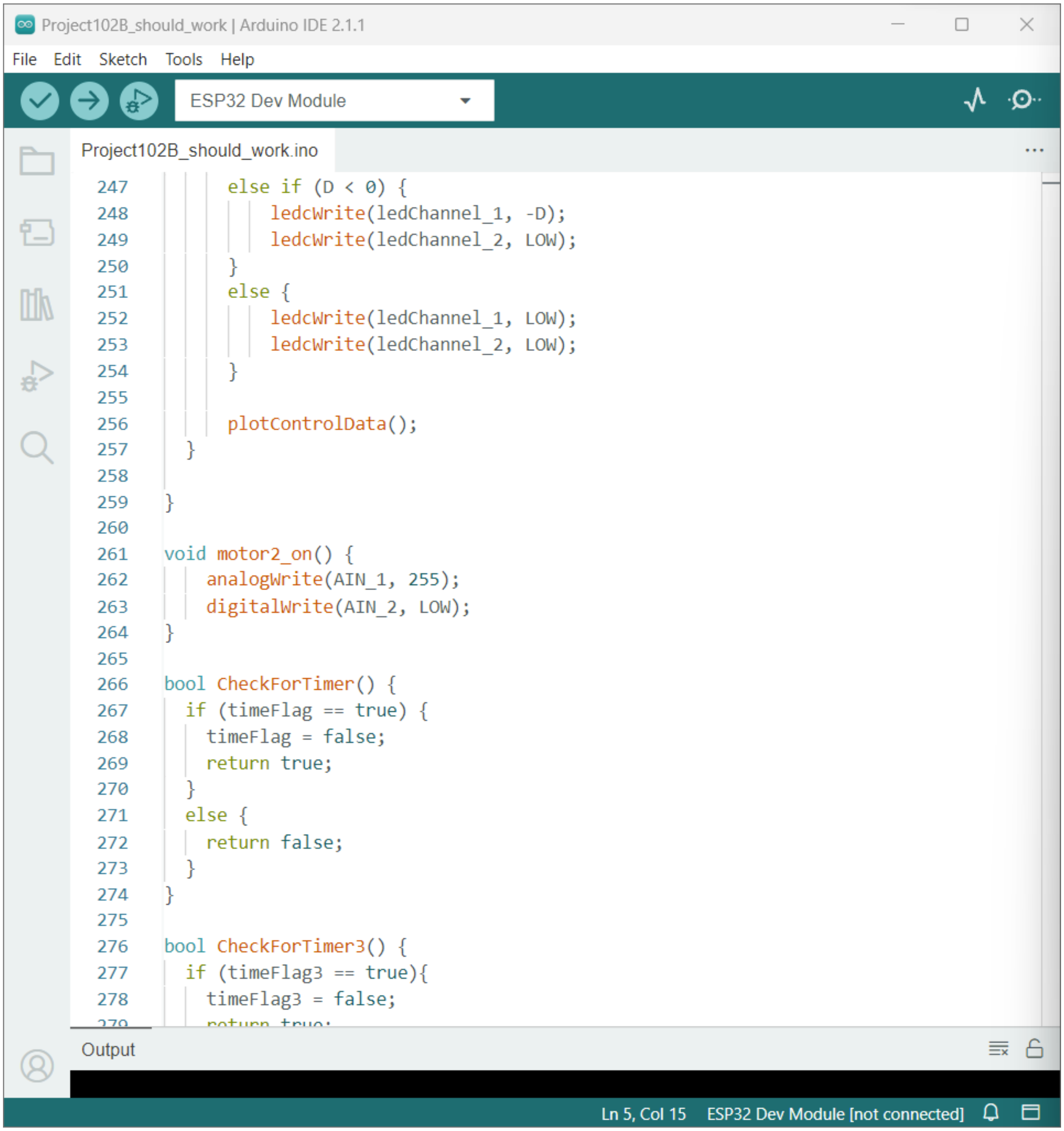
```
    }
  else if (potReading > 1365 && potReading < 1365*2){
      if (deltaT) {
        m = 2800/6;
        fin_er = 2800-m;
      }
    }
    else if (potReading > 1365*2 && potReading < 1365*3){
     if (deltaT) {
        m = 2800/8;
        fin_er = 2800-m;
      }
    }

  switch(state){
    case 0:
      ledcWrite(ledChannel_3, LOW); //Dealing motor off
      ledcWrite(ledChannel_4, LOW);
      motor2_on(); //Turn on shooting motor
      timerStart(timer3);
      if (CheckForTimer3() == true) {
        state = 1;
      }
    case 1:
      ledcWrite(ledChannel_3, 140); //Turn on dealing motor
      ledcWrite(ledChannel_4, LOW);
      timerStart(timer3);
      if (CheckForTimer3() == true){
        state = 2;
      }
      break;

    case 2:
```


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
No Notifications
Ln 5, Col 15
ESP32 Dev Module [not connected]

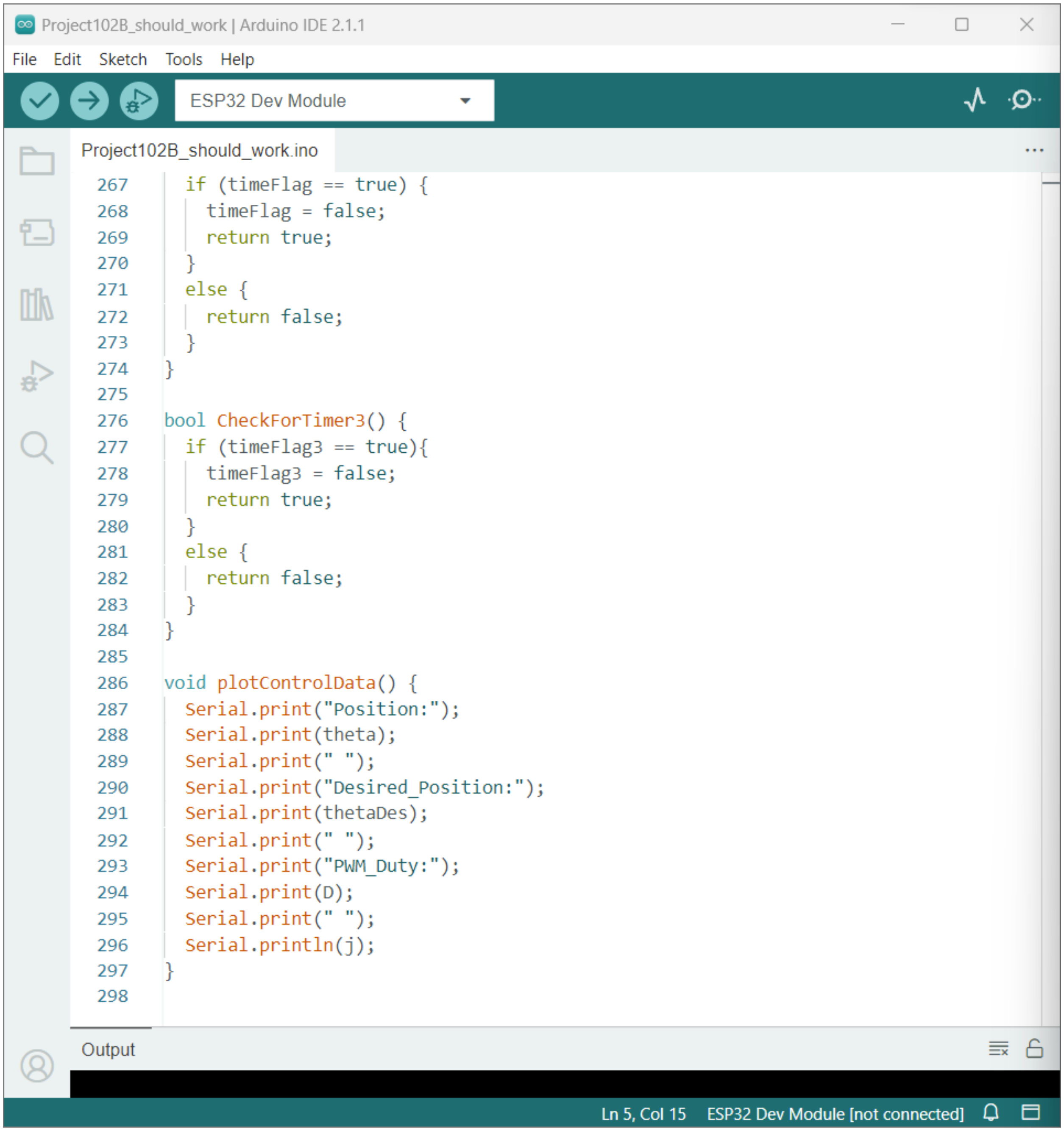
```
    case 2:
      ledcWrite(ledChannel_3, LOW); //Turn off dealing motor
      ledcWrite(ledChannel_4, LOW);
      timerStart(timer2);
      if (CheckForTimer() == true){
        if(abs(theta-fin_er) < 150){
          n = 0;
          state = 4;
        }
        else {
          n = theta + m;
          state = 3;
        }
      }

      break;


    case 3:
      Rotating(); //Rotate to next player
      timerStart(timer2);
      if (abs(theta-thetaDes) < 150 && CheckForTimer() == true){
          state = 0;
      }
      break;

    case 4:
        Rotating(); //Rotate back to the initial position
        break;
  }

}
```


Project102B_should_work | Arduino IDE 2.1.1
File Edit Sketch Tools Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15 ESP32 Dev Module [not connected]

```
//Other functions
void Rotating() {
  if (deltaT) {
    portENTER_CRITICAL(&timerMux1);
    deltaT = false;
    portEXIT_CRITICAL(&timerMux1);

    theta += count;
    thetaDes = n;

    //A6 CONTROL SECTION
    //CHANGE THIS SECTION FOR P AND PI CONTROL
    err = thetaDes-theta;
    P = Kp*err;
    sumerr = sumerr + err;
    D = Kp*(err + (Ki*sumerr));

    //Ensure that you don't go past the maximum possible command
    if (D > MAX_PWM_VOLTAGE) {
      D = MAX_PWM_VOLTAGE;
    }
    else if (D < -MAX_PWM_VOLTAGE) {
      D = -MAX_PWM_VOLTAGE;
    }

    //Map the D value to motor directionality
    //FLIP ENCODER PINS SO SPEED AND D HAVE SAME SIGN
    if (D > 0) {
      ledcWrite(ledChannel_1, LOW);
      ledcWrite(ledChannel_2, D);
    }
    else if (D < 0) {
```


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15
ESP32 Dev Module [not connected]

```
      else if (D < 0) {
          ledcWrite(ledChannel_1, -D);
          ledcWrite(ledChannel_2, LOW);
      }
      else {
          ledcWrite(ledChannel_1, LOW);
          ledcWrite(ledChannel_2, LOW);
      }

      plotControlData();
  }

}

void motor2_on() {
    analogWrite(AIN_1, 255);
    digitalWrite(AIN_2, LOW);
}

bool CheckForTimer() {
  if (timeFlag == true) {
    timeFlag = false;
    return true;
  }
  else {
    return false;
  }
}

bool CheckForTimer3() {
  if (timeFlag3 == true){
    timeFlag3 = false;
    return true;
```


Project102B_should_work | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
ESP32 Dev Module
Project102B_should_work.ino
Output
Ln 5, Col 15
ESP32 Dev Module [not connected]

```
  if (timeFlag == true) {
    timeFlag = false;
    return true;
  }
  else {
    return false;
  }
}

bool CheckForTimer3() {
  if (timeFlag3 == true){
    timeFlag3 = false;
    return true;
  }
  else {
    return false;
  }
}

void plotControlData() {
  Serial.print("Position:");
  Serial.print(theta);
  Serial.print(" ");
  Serial.print("Desired_Position:");
  Serial.print(thetaDes);
  Serial.print(" ");
  Serial.print("PWM_Duty:");
  Serial.print(D);
  Serial.print(" ");
  Serial.println(j);
}

```