```
//Pin definitions
const int selectPin = P4_6;
const int XoutPin = P5_0;
const int buzzer1 = P5_2;

//initialize variables
int forwardX = 0;
int backwardX = 0;
int jump = 0;
int cnt = 0;
int timer = 0;

OneMsTaskTimer_t timerTask = {250,  playerActionTimerISR, 0, 0};

void setupPlayerActions(){
  Serial.begin(9600);
  playerActionState = PlayActionInit;

  //Timer Interrupt SetUp
  OneMsTaskTimer::add(&timerTask);
  OneMsTaskTimer::start();

  //Joystick setup
  pinMode(selectPin, INPUT);
}

//loop through player actions by reading the sensors and then
//going to the state machine for actual actions
void loopPlayerActions(){
  while(timer ==0) {delay(10);}
  timer = 0;
  readSensors();
  playerActionTickFunc();
  delay(10);
}

void playerActionTickFunc(){

    //State Transitions
  switch (playerActionState){
        case PlayActionInit:
        playerActionState = GameStart;
        break;
```

```
  case GameStart:
    playerActionState = WaitingForAction;
    break;

  case WaitingForAction:

    if (jump == 1){
        playerActionState = Jump;
        cnt = 0;
        jump = 0;
    }
    else if (forwardX == 1){
      playerActionState = MoveForward;
      forwardX = 0;
    }
    else if (backwardX == 1){
      playerActionState = MoveBack;
      backwardX = 0;
    }
    break;

  case MoveForward:
    playerActionState = WaitingForAction;
    break;

  case MoveBack:
    playerActionState = WaitingForAction;
    break;

  case Jump:
    //check if hero has been in air for long enough before going back to next state
    if (cnt >= 3){
        playerActionState = WaitingForAction;
        heroPos.y = 1;
    }
    //increment for jump to be held
    cnt++;
    break;

 }


//PlayActionInit, GameStart, WaitingForAction, MoveForward, MoveBack, Jump
```

```
//State Actions
switch (playerActionState){
    case GameStart:
      heroPos.x = 0;
      heroPos.y = 1;
      lcd.setCursor(0, 0);
      lcd.print("                ");
      lcd.setCursor(0, 1);
      lcd.print("                ");
      break;

    case WaitingForAction:
       break;

    case MoveForward:
        (heroPos.x)++;
        break;

    case MoveBack:
        (heroPos.x)--;
        break;

    case Jump:
        heroPos.y = 0;
        break;
  }
}

void readSensors(){
  //Read in the stick position
  int joystickXPos = analogRead(XoutPin);
  Serial.println(joystickXPos);
  //strange threshold values to account for drift
  if (joystickXPos > 977){
    forwardX = 1;
  }
  else if (joystickXPos < 965){
    backwardX = 1;
  }
}

void playerActionTimerISR(){
  int jumpVal = analogRead(selectPin);
  //detect jump
```

```
  if (jumpVal < 600){
    jump = 1;
    //make sound when character jumps
    tone(buzzer1, 1200, 100);
  }
  timer = 1;
}
```