//IN MAIN PAGE……

```
#include <PS4Controller.h>
#include <ESP32_Servo.h>
#include <HUSKYLENS.h>
#include <Wire.h>
HUSKYLENS huskylens;
#define HUSKYLENS_ADDRESS 0x32
//HUSKYLENS green line >> SDA; blue line >> SCL
//int ID0 = 0; //not learned results. Grey result on HUSKYLENS screen
int ID1 = 1;  //first learned results. colored result on HUSKYLENS screen
//int ID2 = 2; //second learned results. colored result on HUSKYLENS screen
// and so on.....

Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo6;
Servo myservo8;
Servo myservo9;


void printResult(HUSKYLENSResult result);

int connected = 2;
int LY = 1500;
int RY = 1500;
int left_error;
int right_error;
int golf = 18;
int basketball = 4;
int gripper = 19;
int gripperZ = 5;
int nut = 0;
boolean options = true;
```

IN VOID SETUP……

```
  void setup() {
  Serial.begin(115200);
  PS4.begin("64:B7:08:41:57:58");

  myservo1.attach(5);  //gripperZ
```

```
  myservo2.attach(16);  //LY
  myservo3.attach(0);   //nut
  myservo4.attach(17);  //RY
  myservo6.attach(18); //golf
  myservo8.attach(4);  //basketball
  myservo9.attach(19); //gripper



  Serial.println("Ready.");
  pinMode(connected, OUTPUT);
  digitalWrite(connected, LOW);



  Wire.begin();
  while (!huskylens.begin(Wire)) {
    Serial.println(F("Begin failed!"));
    Serial.println(F("1.Please recheck the \"Protocol Type\" in HUSKYLENS
(General Settings>>Protocol Type>>I2C)"));
    Serial.println(F("2.Please recheck the connection."));
    delay(100);
  }

  // Uncomment one of the following code to switch the algorithm on HUSKYLENS:
  // huskylens.writeAlgorithm(ALGORITHM_FACE_RECOGNITION);
   huskylens.writeAlgorithm(ALGORITHM_OBJECT_TRACKING);
   //huskylens.writeAlgorithm(ALGORITHM_OBJECT_RECOGNITION);
  // huskylens.writeAlgorithm(ALGORITHM_LINE_TRACKING);
   //huskylens.writeAlgorithm(ALGORITHM_COLOR_RECOGNITION);
  //huskylens.writeAlgorithm(ALGORITHM_TAG_RECOGNITION);
  }
```

//IN VOID LOOP

```
void loop() {
  // Below has all accessible outputs from the controller
  if (PS4.isConnected()) {
    digitalWrite(connected, HIGH);
  } else {
    digitalWrite(connected, LOW);
  }
```

```
//LY----------------------------------------------------------------LY

if (PS4.LStickY() > 10 || PS4.LStickY() < -10) {

  LY = map(PS4.LStickY(), -128, 128, 1000, 2000);
  myservo2.writeMicroseconds(LY);
  //Serial.printf("Left Stick y at %d\n", (LY))
}

else {
  LY = 1500;
  myservo2.writeMicroseconds(LY);
}

//RY----------------------------------------------------------------RY

if (PS4.RStickY() > 10 || PS4.RStickY() < -10) {

  RY = map(PS4.RStickY(), -128, 128, 1000, 2000);
  myservo4.writeMicroseconds(RY);
  //Serial.printf("Right Stick y at %d\n", (RY));
}

else {
  RY = 1500;
  myservo4.writeMicroseconds(RY);
}



if (PS4.Right()) {
    myservo3.writeMicroseconds(2000);
} else if (PS4.Left()) {
  myservo3.writeMicroseconds(-2000);
   }else {
    nut = 1500;
    myservo3.writeMicroseconds(nut);
  }

if (PS4.Down()) {
  myservo8.writeMicroseconds(2000);
} else if (PS4.Up()) {
  myservo8.writeMicroseconds(-2000);
} else {
```

```
  basketball = 1500;
  myservo8.writeMicroseconds(basketball);
}

//  if (PS4.Cross()) Serial.println("Cross Button");
if (PS4.Triangle()) {
  myservo6.writeMicroseconds(2000);
} else {
  golf = 1500;
  myservo6.writeMicroseconds(golf);
}
//if (PS4.UpRight())
// if (PS4.DownRight()) Serial.println("Down Right");
// if (PS4.UpLeft()) Serial.println("Up Left");
// if (PS4.DownLeft()) Serial.println("Down Left");
if (PS4.R1()) {
  myservo9.writeMicroseconds(2000);
} else if (PS4.L1()) {
  myservo9.writeMicroseconds(-2000);
   }else {
    gripper = 1500;
    myservo9.writeMicroseconds(gripper);
  }
if (PS4.L3()) Serial.println("L3 Button");
if (PS4.R3()) Serial.println("R3 Button");

if (PS4.PSButton()) Serial.println("PS Button");
if (PS4.Touchpad()) Serial.println("Touch Pad Button");

if (PS4.L2()) {
    myservo1.writeMicroseconds(2000);
} else if (PS4.R2()) {
  myservo1.writeMicroseconds(-2000);
   }else {
    gripperZ = 1500;
    myservo1.writeMicroseconds(gripperZ);
  }


if (PS4.LStickX()) {
  Serial.printf("Left Stick x at %d\n", PS4.LStickX());
}
if (PS4.LStickY()) {
  Serial.printf("Left Stick y at %d\n", PS4.LStickY());
}
```

```
  if (PS4.RStickX()) {
    Serial.printf("Right Stick x at %d\n", PS4.RStickX());
  }
  if (PS4.RStickY()) {
    Serial.printf("Right Stick y at %d\n", PS4.RStickY());
  }

  //if (PS4.Charging()) Serial.println("The controller is charging");
  // if (PS4.Audio()) Serial.println("The controller has headphones attached");
  //  if (PS4.Mic()) Serial.println("The controller has a mic attached");

  //Serial.printf("Battery Level : %d\n", PS4.Battery());

 // Serial.println();
  // This delay is to make the output more human readable
  // Remove it when you're not trying to see the output
  //delay(75);


        //SHARE-----------------------------------------------------SHARE

      if (PS4.Share() && (!options)) {
        Serial.println("EXIT HUSKY");
      options = !options;
       delay(250);
        return;
      }

      //OPTIONS---------------------------------------------------OPTIONS

      if (PS4.event.button_down.options) {
        options = !options;
      }
      if (!options) {
        HL();
      }
    }
```

// IN HUSKY……

```
//HUSKY_LENS-------------------------------------------------------------------
-----------------------------------------
void HL() {

  if (!huskylens.request()) Serial.println(F("Fail to request data from
HUSKYLENS, recheck the connection!"));
  // if (huskylens.requestBlocks())           //request only blocks from
HUSKYLENS
  // if (huskylens.requestArrows())           //request only arrows from
HUSKYLENS
  // if (huskylens.requestLearned())          //request blocks and arrows tangged
ID != 0 from HUSKYLENS
  // if (huskylens.requestBlocksLearned())    //request blocks tangged ID != ID0
from HUSKYLENS
  // if (huskylens.requestArrowsLearned())    //request arrows tangged ID != ID0
from HUSKYLENS
  // if (huskylens.request(ID1))              //request blocks and arrows tangged
ID == ID1 from HUSKYLENS
  // if (huskylens.requestBlocks(ID1))        //request blocks tangged ID == ID1
from HUSKYLENS
  // if (huskylens.requestArrows(ID1))        //request arrows tangged ID == ID1
from HUSKYLENS
  // if (huskylens.request(ID2))              //request blocks and arrows tangged
ID == ID2 from HUSKYLENS
  // if (huskylens.requestBlocks(ID2))        //request blocks tangged ID == ID2
from HUSKYLENS
  // if (huskylens.requestArrows(ID2))        //request arrows tangged ID == ID2
from HUSKYLENS

  else if (!huskylens.isLearned()) Serial.println(F("Nothing learned, press learn
button on HUSKYLENS to learn one!"));
  else if (!huskylens.available()) Serial.println(F("No block or arrow appears on
the screen!"));
  else {
    //Serial.println(F("############"));
    while (huskylens.available()) {
      HUSKYLENSResult result = huskylens.read();
      printResult(result);

      //If vision box is to the left, speed up right side

      if (result.xCenter < 150) {


        left_error = (160 - result.xCenter);
```

```
        if (result.height < 100) {  //If sharp corner is approached, pivot in
place

          myservo2.writeMicroseconds(1850);  //LEFT
          myservo4.writeMicroseconds(1500);  //RIGHT

        } else {

          myservo2.writeMicroseconds(1750);  //LEFT
          myservo4.writeMicroseconds(1350);  //RIGHT

          Serial.println("RIGHT");

          printResult(result);
        }

        //If vision box is to the right, speed up the left side

      } else if (result.xCenter > 170) {

        if (result.height < 100) {  //If sharp corner is approached, pivot in
place

          myservo2.writeMicroseconds(1500);  //LEFT
          myservo4.writeMicroseconds(1150);  //RIGHT

        } else {

          myservo2.writeMicroseconds(1650);  //LEFT
          myservo4.writeMicroseconds(1250);  //RIGHT

          Serial.println("LEFT");

          printResult(result);
        }

      } else if ((result.xCenter < 170) && (result.xCenter > 150)) {

        myservo2.writeMicroseconds(1700);  //LEFT
        myservo4.writeMicroseconds(1300);  //RIGHT

        Serial.println("CENTER");
```

```
        printResult(result);
      }
    }
  }
}



void printResult(HUSKYLENSResult result) {
  if (result.command == COMMAND_RETURN_BLOCK) {
    Serial.println(String() + F("Block:xCenter=") + result.xCenter +
F(",yCenter=") + result.yCenter + F(",width=") + result.width + F(",height=") +
result.height + F(",ID=") + result.ID);
  }  //else if (result.command == COMMAND_RETURN_ARROW) {
  //Serial.println(String() + F("Arrow:xOrigin=") + result.xOrigin +
F(",yOrigin=") + result.yOrigin + F(",xTarget=") + result.xTarget +
F(",yTarget=") + result.yTarget + F(",ID=") + result.ID);
  // }
  else {
    Serial.println("Object unknown!");
  }
}
```