LAMPIRAN A
PROGRAM ARDUINO ALAT

```
#include <PS2X_lib.h>  //for v1.6
#define A 4// L PWM 1
#define B 3 // R PWM 1
#define C 2 // PWM 1
#define D 7 // L PWM 2
#define E 6 // R PWM 2
#define F 5 // PWM 2
#define G 45
#include <Wire.h>
float Tegangan;
float BacaTegangan;

/*
  right now, the library does NOT support hot pluggable controllers, meaning
  you must always either restart your Arduino after you conect the controller,
  or call config_gamepad(pins) again after connecting the controller.
*/
// create PS2 Controller Class
PS2X ps2x;
int error = 0;
byte type = 0;
byte vibrate = 0;

void setup()
```

```
  Serial.begin(57600);
  // CHANGES for v1.6 HERE!!! **PAY ATTENTION*
  // setup pins and settings:  GamePad(sck52, miso,ss, mosi,
Pressures?, Rumble?) check for error SS53 CLK52 MISO51
MOSI50
  error = ps2x.config_gamepad(52,51,53,50, true, true);
pinMode (A,OUTPUT);
pinMode (B,OUTPUT);
pinMode (C,OUTPUT);
pinMode (D,OUTPUT);
pinMode (E,OUTPUT);
pinMode (F,OUTPUT);
pinMode (G,OUTPUT);
  if(error == 0)
  {
    Serial.println("Found Controller, configured successful");
    Serial.println("Try out all the buttons, X will vibrate the
controller, faster as you press harder;");
    Serial.println("holding L1 or R1 will print out the analog
stick values.");
    Serial.println("Go to www.billporter.info for updates and
to report bugs.");
  }
  else if(error == 1)
    Serial.println("No controller found, check wiring, see
readme.txt to enable debug. visit www.billporter.info for
troubleshooting tips");
  else if(error == 2)
    Serial.println("Controller found but not accepting
commands. see readme.txt to enable debug. Visit
www.billporter.info for troubleshooting tips"
```

```
  else if(error == 3)
    Serial.println("Controller refusing to enter Pressures
mode, may not support it. ");

  // Serial.print(ps2x.Analog(1), HEX);
  type = ps2x.readType();
  switch(type)
  {
    case 0:
      Serial.println("Unknown Controller type");
      break;
    case 1:
      Serial.println("DualShock Controller Found");
      break;
    case 2:
      Serial.println("GuitarHero Controller Found");
      break;
  }
}

void loop()
{
  /*
    You must Read Gamepad to get new values
    Read GamePad and set vibration values
    ps2x.read_gamepad(small motor on/off, larger motor
strenght from 0-255)
    if you don't enable the rumble, use ps2x.read_gamepad();
with no values
    you should call this at least once a second
  */
  // skip loop if no controller found
```

```
BacaTegangan=analogRead(3);
Tegangan=((BacaTegangan*0.00489)*5);
Serial.print(Tegangan);
Serial.println("V");
if (Tegangan > 11){
  digitalWrite(G,LOW);
  }
if (Tegangan < 11){
  digitalWrite(G,HIGH);
  }
  if(error == 1)
    return;

  if(type == 2)
  {
    // Guitar Hero Controller
    // read controller
    ps2x.read_gamepad();

    if(ps2x.ButtonPressed(GREEN_FRET))
      Serial.println("Green Fret Pressed");
    if(ps2x.ButtonPressed(RED_FRET))
      Serial.println("Red Fret Pressed");
    if(ps2x.ButtonPressed(YELLOW_FRET))
      Serial.println("Yellow Fret Pressed");
    if(ps2x.ButtonPressed(BLUE_FRET))
      Serial.println("Blue Fret Pressed");
    if(ps2x.ButtonPressed(ORANGE_FRET))
      Serial.println("Orange Fret Pressed");

    if(ps2x.ButtonPressed(STAR_POWER))
      Serial.println("Star Power Command");
```

```
    // will be TRUE as long as button is pressed
    if(ps2x.Button(UP_STRUM))
      Serial.println("Up Strum");
    if(ps2x.Button(DOWN_STRUM))
      Serial.println("DOWN Strum");

    // will be TRUE as long as button is pressed
    if(ps2x.Button(PSB_START))
      Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
      Serial.println("Select is being held");

    // print stick value IF TRUE
    if(ps2x.Button(ORANGE_FRET))
    {
      Serial.print("Wammy Bar Position:");
      Serial.println(ps2x.Analog(WHAMMY_BAR), DEC);
    }
  }
  else
  {
    // DualShock Controller
    // read controller and set large motor to spin at 'vibrate'
speed
    ps2x.read_gamepad(false, vibrate);

    // will be TRUE as long as button is pressed
    if(ps2x.Button(PSB_START))
      Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
      Serial.println("Select is being held");
```

```
 // will be TRUE as long as button is pressed
    if(ps2x.Button(PSB_PAD_UP))
    {
      Serial.print("Up held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_UP), DEC);
  }
    }
    if(ps2x.Button(PSB_PAD_RIGHT))
    {
      Serial.print("Right held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_RIGHT),
DEC);
    }
    if(ps2x.Button(PSB_PAD_LEFT))
    {
      Serial.print("LEFT held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_LEFT), DEC);
    }
    if(ps2x.Button(PSB_PAD_DOWN))
    {
      Serial.print("DOWN held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_DOWN),
DEC);
    }

    // this will set the large motor vibrate speed based on how
hard you press the blue (X) button
    vibrate = ps2x.Analog(PSAB_BLUE);

    // will be TRUE if any button changes state (on to off, or
off to on)
    if (ps2x.NewButtonState())
```

```
  {
   if(ps2x.Button(PSB_L3))
     Serial.println("L3 pressed");
   if(ps2x.Button(PSB_R3))
     Serial.println("R3 pressed");
   if(ps2x.Button(PSB_L2))
     Serial.println("L2 pressed");
   if(ps2x.Button(PSB_R2))
     Serial.println("R2 pressed");
   if(ps2x.Button(PSB_GREEN))
     Serial.println("Triangle pressed");
  }

  // will be TRUE if button was JUST pressed
  if(ps2x.NewButtonState(PSB_RED)){
  Serial.println("Circle just pressed");
}
  // will be TRUE if button was JUST released
  if(ps2x.NewButtonState(PSB_PINK)){
    Serial.println("Square just pressed");
   ;
 }

  // will be TRUE if button was JUST pressed OR released
  if(ps2x.NewButtonState(PSB_BLUE)){
    Serial.println("X just changed");
}
  // print stick values if either is TRUE
  if(ps2x.Analog(PSS_LY) || ps2x.Analog(PSS_LX))
  {
    Serial.print("Stick Values:");
    // Left stick, Y axis. Other options: LX, RY, RX
```

```
      Serial.print(ps2x.Analog(PSS_LY), DEC);
      Serial.print(",");
      Serial.print(ps2x.Analog(PSS_LX), DEC);
      Serial.print(",");
      Serial.print(ps2x.Analog(PSS_RY), DEC);
      Serial.print(",");
      Serial.println(ps2x.Analog(PSS_RX), DEC);
   }
   if (ps2x.Analog(PSS_LY)>128){ // mundur pelan
      digitalWrite(A,HIGH);
      digitalWrite(B,LOW);
      analogWrite(C,100);
      digitalWrite(D,HIGH);
      digitalWrite(E,LOW);
      analogWrite(F,100);}
 if (ps2x.Analog(PSS_LY)==255){ // mundur cepat
      digitalWrite(A,HIGH);
      digitalWrite(B,LOW);
      digitalWrite(C,255);
      digitalWrite(D,HIGH);
      digitalWrite(E,LOW);
      analogWrite(F,255);
     }
      if (ps2x.Analog(PSS_LY)==128){ // stop
      digitalWrite(A,LOW);
      digitalWrite(B,LOW);
      analogWrite(C,0);
      digitalWrite(D,LOW);
      digitalWrite(E,LOW);
      analogWrite(F,0);
       }
        if (ps2x.Analog(PSS_LY)<128) // maju pelan
```

```
   {
     digitalWrite(A,LOW);
     digitalWrite(B,HIGH);
     analogWrite(C,100);
     digitalWrite(D,LOW);
     digitalWrite(E,HIGH);
     analogWrite(F,100);}
if (ps2x.Analog(PSS_LY)==0){ // maju cepat
     digitalWrite(A,LOW);
     digitalWrite(B,HIGH);
     digitalWrite(C,255);
     digitalWrite(D,LOW);
     digitalWrite(E,HIGH);
     analogWrite(F,255);
   }
     if (ps2x.Analog(PSS_LX)==255){ // belok kanan
     digitalWrite(A,LOW);
     digitalWrite(B,HIGH);
     digitalWrite(C,255);
     digitalWrite(D,LOW);
     digitalWrite(E,LOW);
     analogWrite(F,0);}
      if (ps2x.Analog(PSS_LX)==0){ // belok kiri
     digitalWrite(A,LOW);
     digitalWrite(B,LOW);
     digitalWrite(C,0);
     digitalWrite(D,LOW);
     digitalWrite(E,HIGH);
     analogWrite(F,255);}
  delay(50);
}
```