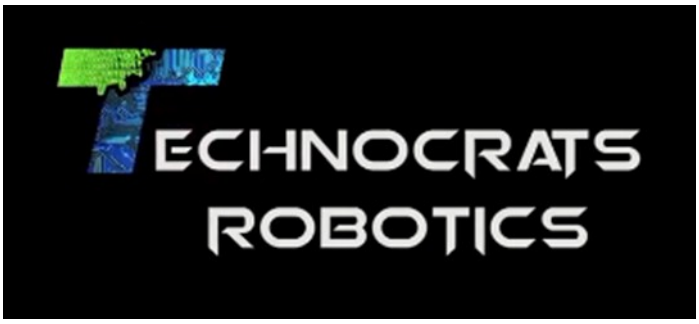


VACATION TASKS

SET-1

Types of Communication Protocols

-

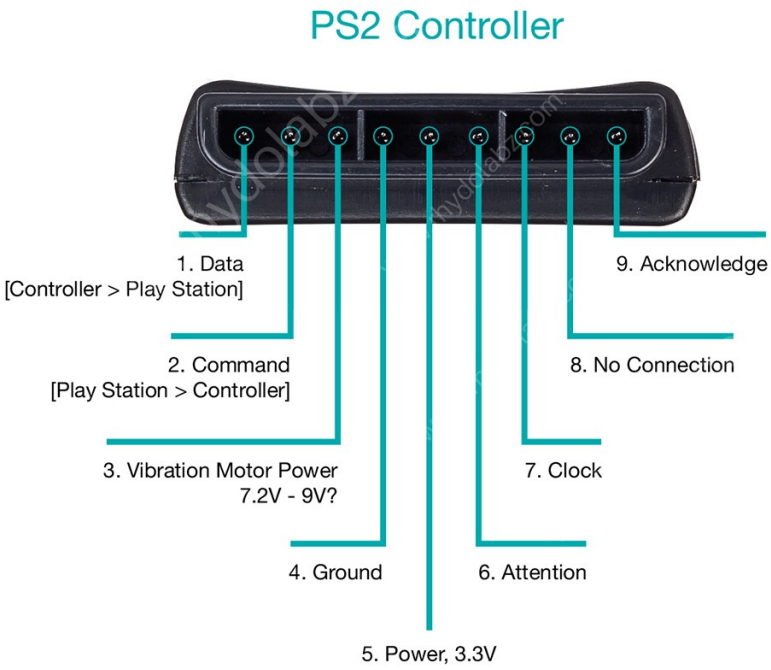
Tanush Biju

Interfacing PS2 Wireless Controller With Arduino

The PS2 wireless controller is a standard controller for the PlayStation 2 and is identical to the original DualShock controller for the PlayStation console. It features twelve analog (pressure-sensitive) buttons (X, O, Π, Δ, L1, R1, L2, R2, Up, Down, Left and Right), five digital button (L3, R3 Start, Select and the analog mode button) and two analog sticks.

The controller also features two vibration motors, the left one being larger and more powerful than the one on the right. It is powered by two AAA batteries. It communicates with the console using 2.4 GHz RF protocol.

PS2 Receiver Pin Out:

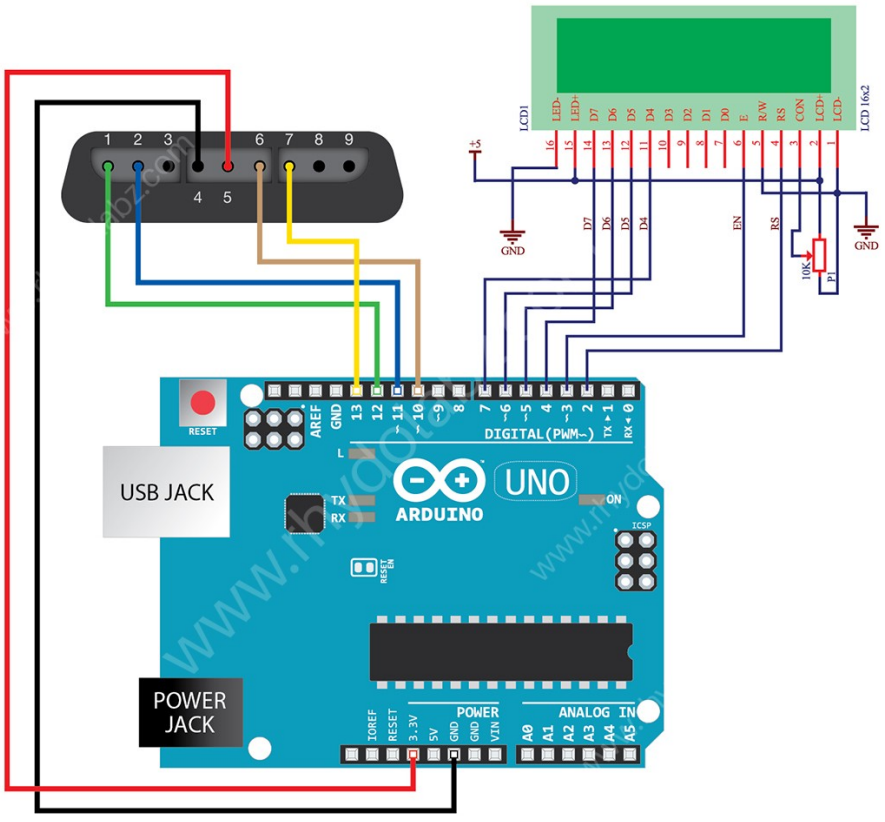


1. DATA: This is the data line from Controller to PS2. This is an open collector output and requires a pull-up resistor (1 to 10k, maybe more). (A pull-up resistor is needed because the controller can only connect this line to ground; it can't actually put voltage on the line).
2. COMMAND: This is the data line from PS2 to Controller.
3. VIBRATION MOTOR POWER
4. GND: Ground
5. VCC: VCC can vary from 5V down to 3V .
6. ATT: ATT is used to get the attention of the controller. This line must be pulled low before each group of bytes is sent / received, and then set high again afterwards. This pin consider as "Chip Select" or "Slave Select" line that is used to address different controllers on the same bus.

7. CLK: 500kH/z, normally high on. The communication appears to be SPI bus.
8. Not Connected
9. ACK: Acknowledge signal from Controller to PS2. This normally high line drops low about 12us after each byte for half a clock cycle, but not after the last bit in a set. This is an open collector output and requires a pull-up resistor (1 to 10k, maybe more). PS2 Signals:

PS2 wireless controller communicates with Arduino using a protocol that is basically SPI. The play station sends a byte at the same time as it receives one (full duplex) via serial communication. There's a clock (SCK) to synchronize bits of data across two channels: DATA and CMD. Additionally, there's an "Attention" (ATT) channel which tells the slave whether or not it is "active" and should listen to data bits coming across the CMD channel, or send data bits across the DATA channel (Reasonably, only one slave device should be active at a time). The PlayStation 2 actually uses this plus an additional line that is not specifically part of the SPI protocol – an "Acknowledge" (ACK) line.

The clock is held high until a byte is to be sent. It then drops low (active low) to start 8 cycles during which data is simultaneously sent and received. The logic level on the data lines is changed by the transmitting device on the falling edge of clock. This is then read by the receiving device on the rising edge allowing time for the signal to settle. After each Command is received from the controller, that controller needs to pull ACK low for at least one clock cycle. If a selected controller does not ACK the PS2 will assume that there is no controller present. LSBs (least significant bits) are transmitting first.

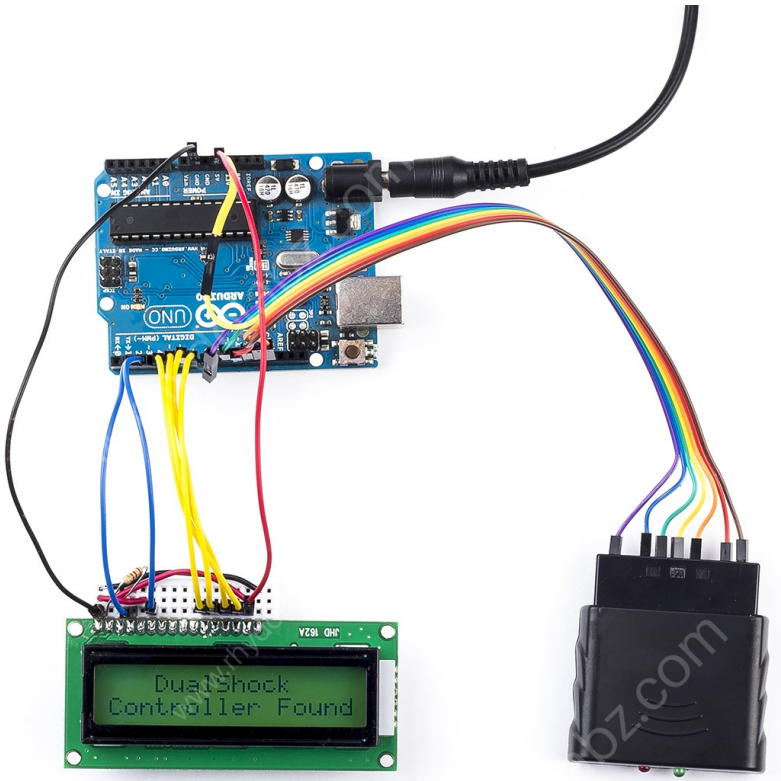


Connection Diagram

Upon each button press the Arduino receives the RF signal on the PS2 receiver and displays the it on the alphanumeric LCD module. We followed the standard PS2 protocol for realizing the communication algorithm, identical to the SPI protocol. Our program on the Arduino detects and reads the button presses only, pressure values are not read. Analog stick state values are continuously displayed on the LCD module.

Connection Details:

The PS2 receiver CLK line and ATT lines are held normally high. The ATT operates like the Slave Select line under SPI. You pull it low to tell the controller you are talking to it and then send it back high once a communications cycle is complete. CMD is the data line to the controller and DATA is the data coming from the controller.



For interfacing, a PS2 library for Arduino is required.(PS2X)

Arduino Code

```
#include <PS2X_lib.h>                    /* PS2 Controller Library;Available Online*/
#include <LiquidCrystal.h>               /* LiquidCrystal Library */
PS2X ps2x;                              /* create PS2 Controller Class*/
byte Type = 0;
byte vibrate = 0;
int RX=0,RY=0,LX=0,LY=0;
LiquidCrystal lcd(2,3,7, 6, 5, 4);          /* initialize the library with the numbers of the interface
pins*/
void setup(){
  lcd.begin(16, 2);                        /* 16X2 lcd display */
  ps2x.config_gamepad(13,11,10,12, true, true); /* setup pins and settings:  GamePad(clock,
command, attention, data, Pressures?, Rumble?) check for error*/
  Type = ps2x.readType();                    /* Reading type of the PS2 Controller */
  if(Type==1){                            /* Type 1 is Duel shock controller */
    lcd.setCursor(0, 0);                   /* Setting display position*/
    lcd.print("   DualShock    ");          /* display if the controller is duel shock*/
    lcd.setCursor(0, 1) ;
    lcd.print("Controller Found");
    delay(1000);
    lcd.clear();
  }
}
void loop(){

  ps2x.read_gamepad(false, vibrate);   /* read controller and set large motor to spin at
'vibrate' speed */
  lcd.setCursor(0, 0);               /* Position the LCD cursor */
  lcd.print("Stick values:   ");      /* Display analog stick values */
  lcd.setCursor(0, 1);
  LY = ps2x.Analog(PSS_LY);         /* Reading Left stick Y axis */
  LX = ps2x.Analog(PSS_LX);         /* Reading Left stick X axis */
  RY = ps2x.Analog(PSS_RY);         /* Reading Right stick Y axis */
  RX = ps2x.Analog(PSS_RX);         /* Reading Right stick X axis */
  if((LY <= 9))                    /* standardize to 3 digit by checking less than 10 */
    lcd.print("00");               /* eg: if LY= 5 then it display as "005" in lcd */
  if((LY >= 9 &&LY <= 99))          /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");                /* eg: if LY= 55 then it display as "055" in lcd */
  lcd.print(LY,DEC);                /* display left analog stick Y axis */
  lcd.print(",");                  /* separate values using comma */
  if((LX <= 9))                    /* standardize to 3 digit by checking less than 10 */
    lcd.print("00");               /* eg: if LX= 5 then it display as "005" in lcd */
  if((LX >= 9 && LX<=99))           /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");                /* eg: if LX= 55 then it display as "055" in lcd */
  lcd.print(LX,DEC);                /* display left analog stick X axis */
  lcd.print(",");                  /* separate values using comma */
  if((RY <= 9))                    /* standardize to 3 digit by checking less than 10 */
```

```
    lcd.print("00");              /* eg: if RY= 5 then it display as "005" in lcd */
  if((RY >= 9 &&RY<=99))           /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");              /* eg: if RY= 55 then it display as "055" in lcd */
  lcd.print(RY,DEC);               /* display Right analog stick Y axis             */
  lcd.print(",");                 /* separate values using comma */
  if((RX <= 9))                    /* standardize to 3 digit by checking less than 10 */
    lcd.print("00");              /* eg: if RX= 5 then it display as "005" in lcd */
  if((RX >= 9 &&RX <= 99))         /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");              /* eg: if RX= 55 then it display as "055" in lcd */
  lcd.print(RX,DEC);               /* display Right analog stick X axis         */
  lcd.print(" ");
  if(ps2x.NewButtonState()) {       /* will be TRUE if any button changes state  */
    lcd.setCursor(0, 0);
    if(ps2x.Button(PSB_START))      /* will be TRUE as long START button is pressed */
      lcd.print("START PRESSED   ");
    if(ps2x.Button(PSB_SELECT))           /* will be TRUE as long SELECT button is pressed
*/
      lcd.print("SELECT PRESSED  ");
    if(ps2x.Button(PSB_PAD_UP))           /* will be TRUE as long as UP button is pressed */
      lcd.print("UP PRESSED      ");
    if(ps2x.Button(PSB_PAD_RIGHT))         /* will be TRUE as long as UP button is pressed
*/
      lcd.print("RIGHT PRESSED   ");
    if(ps2x.Button(PSB_PAD_LEFT))          /* will be TRUE as long as LEFT button is pressed
*/
      lcd.print("LEFT PRESSED    ");
    if(ps2x.Button(PSB_PAD_DOWN))          /* will be TRUE as long as DOWN button is
pressed */
      lcd.print("DOWN PRESSED    ");
    if(ps2x.Button(PSB_L1))               /* will be TRUE as long as L1 button is pressed */
      lcd.print("L1 pressed      ");
    if(ps2x.Button(PSB_R1))               /* will be TRUE as long as R1 button is pressed */
      lcd.print("R1 pressed      ");
    if(ps2x.Button(PSB_L2))               /* will be TRUE as long as L2 button is pressed */
      lcd.print("L2 pressed      ");
    if(ps2x.Button(PSB_R2))               /* will be TRUE as long as R2 button is pressed */
      lcd.print("R2 pressed      ");
    if(ps2x.Button(PSB_L3))               /* will be TRUE as long as L3 button is pressed */
      lcd.print("L3 pressed      ");
    if(ps2x.Button(PSB_R3))               /* will be TRUE as long as R3 button is pressed */
      lcd.print("R3 pressed      ");
    if(ps2x.Button(PSB_GREEN))            /* will be TRUE as long as GREEN/Triangle button
is pressed */
      lcd.print("Triangle pressed");
    if(ps2x.Button(PSB_BLUE))             /* will be TRUE as long as BLUE/CROSS/X button
is pressed */
      lcd.print("X pressed       ");
     if(ps2x.Button(PSB_RED))             /* will be TRUE as long as RED/Circle button is
pressed */
```

```
        lcd.print("Circle pressed  ");
      if(ps2x.Button(PSB_PINK))             /* will be TRUE as long as PINK/Squre button is
pressed */
        lcd.print("Square pressed  ");
      delay(700);
   }
   else;
}
```