

Az-Delivery

Welcome!

Thank you very much for purchasing our AZ-Delivery PS2 Joystick Shield for Arduino. On the following pages, we will introduce you to how to use and setup this handy device.

Have fun!



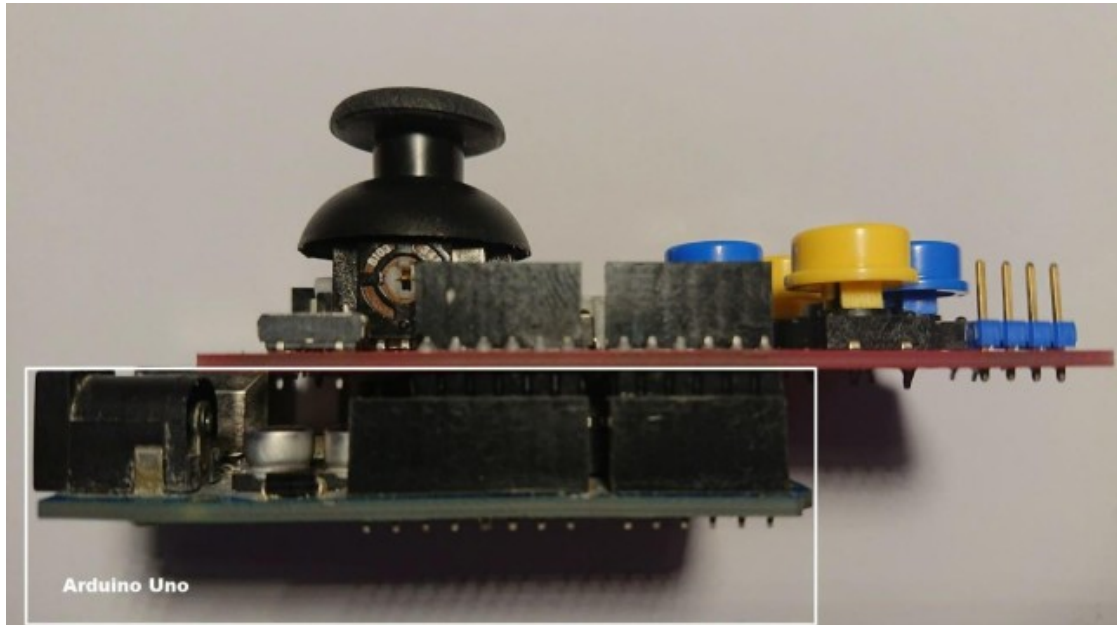


The PS2 Joystick Shield (Funduino) is a plug-in attachment for the Arduino Uno or any other Arduino Uno voltage and port compatible board. It offers a digital control pad with 4 buttons, an analog 2-axis control pad with digital pushbutton and two additional buttons for extended control.

The PS2 Joystick Shield also has various connection options for expansion modules. So it is prepared for the connection of a 4 pole Bluetooth module, a NRF24L01 radio module, a Nokia 5150 display or for the connection of I2C bus elements.

In this manual we focus on the main functions of the shield. The connection of external components is described in detail in another e- book.

Az-Delivery



Now connect your Arduino to your PC. The red LED on your PS2 Joystick Shield should light up. With this you have already created all the necessary technical connections to use the Shield in programs.

To make it easier for you to understand the programming for the shield, we have divided the e-book into 3 parts. The first part deals with the connection of the buttons (digital part) and the second part with the connection of the joystick (analog part) to the Arduino. In the third part we will output the operating status of the buttons and the joystick as text information on the serial interface in a sketch.

Az-Delivery

The output can then be evaluated with your own program on the PC side. In the first part we take a look at the digital part of the PS2 Joystick Shields and with it the wiring and the query of the individual buttons. The buttons on the PS2 Joystick Shield are marked with the letters A to F and K. The button K has a special position, because it can only be operated by pressing lightly on the head of the analog joystick. The other buttons can be operated separately and independently of each other.

The port assignment of the buttons is as follows:

A - (up)	> D2
B - (right)	> D3
C - (down)	> D4
D - (left)	> D5
E - (status taster)	> D6
F - (status taster)	> D7
K - (joystick digital taster)	> D8

Without exception, all push-buttons pull the port to ground when actuated, so that the push-buttons can be seen as active LOW on the programming side. Therefore it is useful to configure the ports as input with activated internal pullup resistor. Therefore in the SETUP part of the code in part 3 the ports are also initialized with the command:

```
pinMode(PIN, INPUT_PULLUP)
```

Az-Delivery

In the second part we now come to the analog part of the PS2 Joystick Shield and thus to the versatile analog joystick. The analog joystick is technically realized via a horizontal (X axis) potentiometer on analog port 0 of the Arduino and a vertical (Y axis) potentiometer on analog port 1 of the Arduino. The movement of the joystick in space is converted into different voltages at the potentiometers. A spring mechanism in the joystick always returns the joystick to the zero position (centre) when not in use. The stick position can be queried by continuously polling the analog converter ports 0 and 1.

Attention!!!

Continuous analog-digital conversions of the processor do not always produce exactly the same values due to internal measurement errors, despite the same position of the control joystick. So the returned value in the zero position can vary between 1 and 5 values.

A switch S1 can be used to switch the voltage from 5.5V to 3.3V for the voltage dividers of the analog control joystick.

Voltage (switch position S1) > Value range

5V > 0..1023

3.3V > 0..662

In this way the resolution of the analog joystick can be changed. When set to 3.3V, the resolution is 663 steps according to the table above and 1024 steps when set to 5V.

Az-Delivery

In the last section of our manual we now come to the Arduino example sketch which outputs the data of the PS2 Joystick Shields via the serial interface of the Arduino:

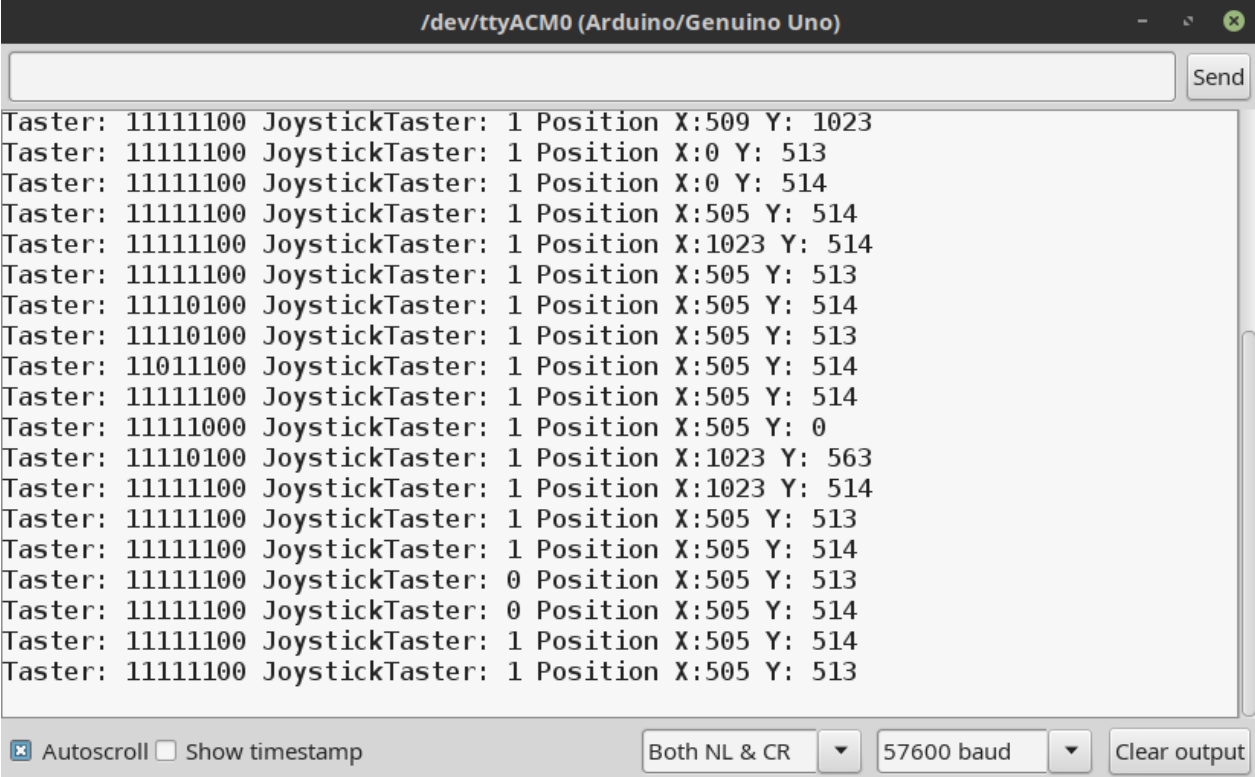
```
// Digital inputs, connected to push buttons
const byte PIN_BUTTON_A = 2;
const byte PIN_BUTTON_B = 3;
const byte PIN_BUTTON_C = 4;
const byte PIN_BUTTON_D = 5;
const byte PIN_BUTTON_E = 6;
const byte PIN_BUTTON_F = 7;
const byte PIN_BUTTON_K = 8;
// X Coordinate is queried via analog port Min:0 / Max: 1023
// A nalog Inputs, connected to the joystiyck coordinate is queried
// via analog port 0 Min:0 / Max: 1023 const byte
const byte PIN_ANALOG_X = 0;
const byte PIN_ANALOG_Y = 1;
byte ButtonStatus = 0;
byte ButtonOLDStatus = 0;
byte JoystickButtonStatus = 1;
byte JoystickButtonOLDStatus = 1;
int XCoord = 0;
int XOLDCoord = 0;
int YCoord = 0;
int YOLDCoord = 0;
void setup() {
    Serial.begin(57600);
    pinMode(PIN_BUTTON_A, INPUT_PULLUP);
    pinMode(PIN_BUTTON_B, INPUT_PULLUP);
    pinMode(PIN_BUTTON_C, INPUT_PULLUP);
    pinMode(PIN_BUTTON_D, INPUT_PULLUP);
    pinMode(PIN_BUTTON_E, INPUT_PULLUP);
    pinMode(PIN_BUTTON_F, INPUT_PULLUP);
    pinMode(PIN_BUTTON_K, INPUT_PULLUP);
}
```

Az-Delivery

```
void loop() {
    ButtonStatus = PIND & 0b11111100;
    // Pin 2 is configured to input with pullup resistor (Port D)
    // Pin 3 is configured to input with pullup resistor. (Port D)
    // Pin 4 is configured to input with pullup resistor. (Port D)
    // Pin 5 is configured to input with pullup resistor. (Port D)
    // Pin 6 is configured to input with pullup resistor. (Port D) .
    // Pin 7 is configured to input with pullup resistor. (Port D)
    // Pin 8 is configured to input with pullup resistor
    // Reading of inputs 2-7 directly via PortD
    // A bitwise AND mask prevents the reading of
    // pins 0 and 1 (serial interface)
    JoystickButtonStatus = digitalRead(PIN_BUTTON_K);
    XCoord = analogRead(PIN_ANALOG_X);
    YCoord = analogRead(PIN_ANALOG_Y);
    if( (ButtonStatus != ButtonOLDStatus)|| (XCoord != XOLDCoord)||
        (YCoord != YOLDCoord)|| (JoystickButtonStatus !=
            JoystickButtonOLDStatus)) {
        delay(100); // Button debounce
        ButtonOLDStatus = ButtonStatus;
        JoystickButtonOLDStatus = JoystickButtonStatus;
        XOLDCoord = XCoord;
        YOLDCoord = YCoord;
        Serial.print("Taster: ");
        Serial.print(ButtonStatus, BIN);
        Serial.print(" JoystickTaster: ");
        Serial.print(JoystickButtonStatus, BIN);
        Serial.print(" Position X:");
        Serial.print(XCoord);
        Serial.print("Y: ");
        Serial.println(YCoord);
    }
}
```

Az-Delivery

Our program generates the following output on the serial interface at 57600 baud rate and S1 switch position 5V.



The screenshot shows a serial terminal window titled "/dev/ttyACM0 (Arduino/Genuino Uno)". The window contains a list of 20 lines of text, each representing a joystick position reading. The text is as follows:

```
Taster: 11111100 JoystickTaster: 1 Position X:509 Y: 1023
Taster: 11111100 JoystickTaster: 1 Position X:0 Y: 513
Taster: 11111100 JoystickTaster: 1 Position X:0 Y: 514
Taster: 11111100 JoystickTaster: 1 Position X:505 Y: 514
Taster: 11111100 JoystickTaster: 1 Position X:1023 Y: 514
Taster: 11111100 JoystickTaster: 1 Position X:505 Y: 513
Taster: 11110100 JoystickTaster: 1 Position X:505 Y: 514
Taster: 11110100 JoystickTaster: 1 Position X:505 Y: 513
Taster: 11011100 JoystickTaster: 1 Position X:505 Y: 514
Taster: 11111100 JoystickTaster: 1 Position X:505 Y: 514
Taster: 11111000 JoystickTaster: 1 Position X:505 Y: 0
Taster: 11110100 JoystickTaster: 1 Position X:1023 Y: 563
Taster: 11111100 JoystickTaster: 1 Position X:1023 Y: 514
Taster: 11111100 JoystickTaster: 1 Position X:505 Y: 513
Taster: 11111100 JoystickTaster: 1 Position X:505 Y: 514
Taster: 11111100 JoystickTaster: 0 Position X:505 Y: 513
Taster: 11111100 JoystickTaster: 0 Position X:505 Y: 514
Taster: 11111100 JoystickTaster: 1 Position X:505 Y: 514
Taster: 11111100 JoystickTaster: 1 Position X:505 Y: 513
```

At the bottom of the window, there are several controls: a checkbox for "Autoscroll" (checked), a checkbox for "Show timestamp" (unchecked), a dropdown menu set to "Both NL & CR", a dropdown menu set to "57600 baud", and a "Clear output" button.

You've done it, you can now use your module for your projects.



Now it is time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

<https://az-delivery.de>

Have Fun!

Impressum

<https://az-delivery.de/pages/about-us>