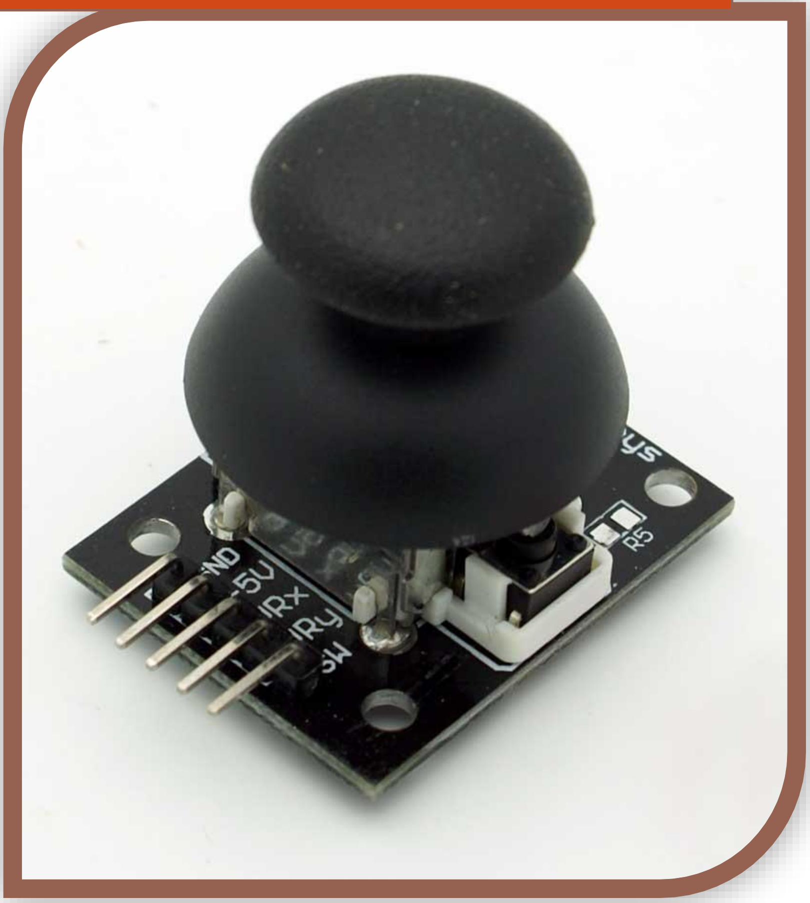
Voltage (Min-Max)	3-5V
# of Pins used	5
Type of pins used	MIXED

MODULES – The JoyStick Module



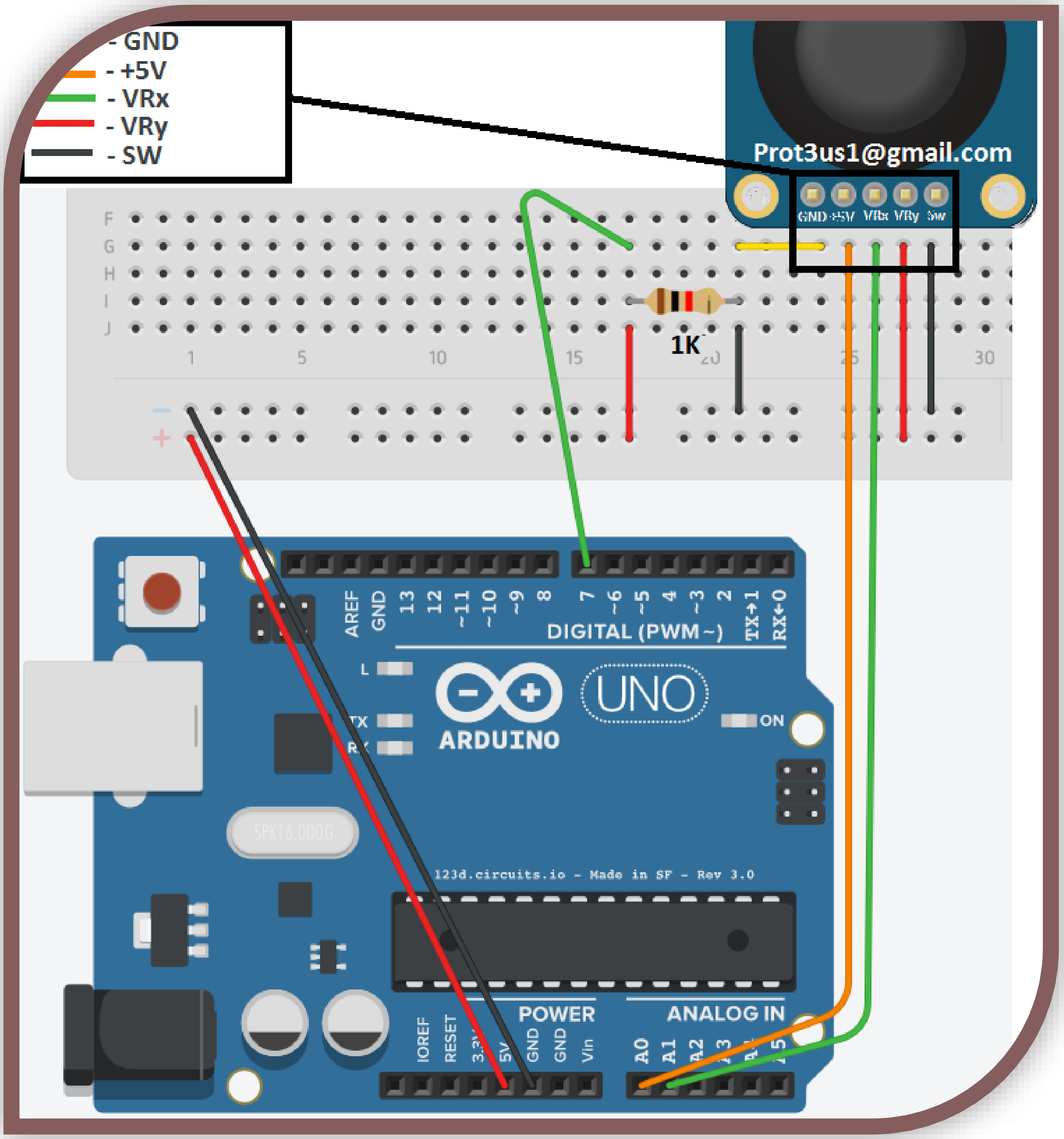
Building with the "JoyStick" module.

By: Tyson Popynick
Aus Electronics Direct

What is the active component in this module?

This module is built around 2 potentiometers and a tactile button. A voltage is passed in to each potentiometer, which then outputs a value that is directly related to the position of the stick. We can then read these values and use them to control our projects. The button will need to be de-bounced as always to prevent multiple readings.

Example Circuit Layout:



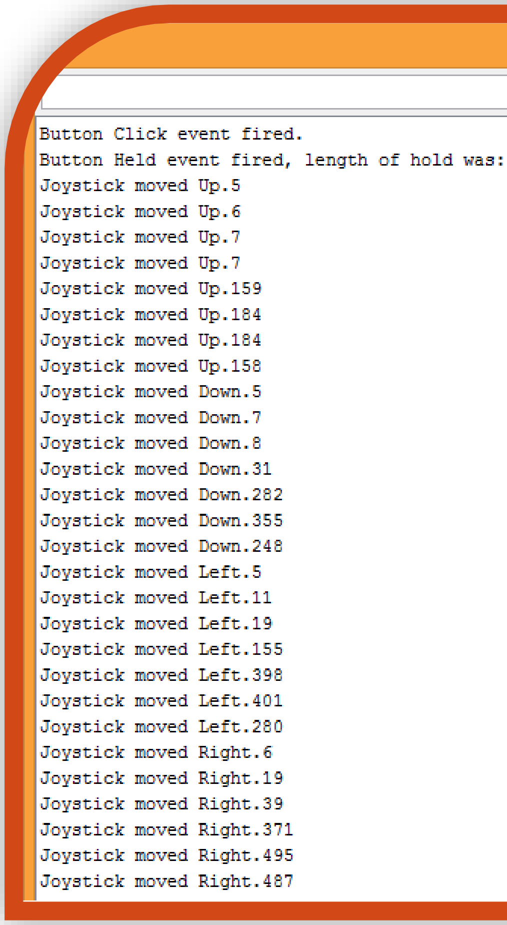
Example Circuit Code:

```
//----Begin Code (copy from here)----
/*
Joystick module example code created by Tyson Popynick (prot3us1@gmail.com) for Aus Electronics Direct.
Outputs are found at the bottom of the sketch, dead zone, null zone and de-bounce time can be set by user.
Debug output verbosity can be set by user.

Gives the following outputs:
Button clicked
Button held + duration
Direction moved + distance moved
*/
//Includes:

//Variables:
//Non changeable variables:
int joyInXPin = A0;            //X-Axis input pin
int joyInYPin = A1;            //Y-Axis input pin
int joyInButton = 7;           //Button input pin
int debounceState = 0;         //Variable to store the debounce state
int isHeld = 0;                //Variable to store the last debounce state
unsigned long lastDebounce = 0;  //Variable to store the last time the button was pressed
int valX = 0;                  //Variable to store the value of the X-axis in
int valY = 0;                  //Variable to store the value of the Y-axis in
int lastX = 0;                 //Variable to hold the last X value
int lastY = 0;                 //Varialble to hold the last Y value

//Changeable Variables, edit the values below to tweak operation.
unsigned long debounceTime = 100; //Minimum time to wait between button state changes
int DEAD_ZONE = 5;             //Joystick dead zone, gives a tiny null area in the middle of the stick to stop ghost values.
int nullX = 500;               //average output of X axis at center
int nullY = 528;               //average output of Y axis at center
//debug settings
int debug_rawXY = 0;           //Should we output raw X/Y Data to serial? (1 = Show / 0 = hide)
int debug_rawButton = 0;       //Should we output raw button Data to serial? (1 = Show / 0 = hide)
int debug_processedButton = 1;   //Should we output processed button Data to serial? (1 = Show / 0 = hide)
int debug_processedJoystick = 1; //Should we output processed joystick data to serial? (1 = show / 0 = hide)

void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
 // put your main code here, to run repeatedly:
 //Joystick input
 lastX = valX;
 lastY = valY;
 valX = analogRead(joyInXPin);
 valY = analogRead(joyInYPin);
 if (valX > nullX && (valX - nullX) >= DEAD_ZONE) {
   //Up
  joyUp(valX - nullX);
 }
 if (valX < nullX && (nullX - valX) >= DEAD_ZONE) {
  //Down
  joyDown(nullX - valX);
 }
 if (valY < nullY && (nullY - valY) >= DEAD_ZONE) {
  //Left
  joyLeft(nullY - valY);
 }
 if (valY > nullY && (valY - nullY) >= DEAD_ZONE) {
  //Right
  joyRight(valY - nullY);
 }
 //Print the values to the Serial Output if we have the debug variable set to 1
 if (debug_rawXY == 1) {
  Serial.print("X Axis: ");
  Serial.print(valX);
  Serial.print(" Y Axis: ");
  Serial.println(valY);
 }
 //Button input
 if (debounceState == 1) {
  if ((millis() - lastDebounce) >= debounceTime) {
    if (digitalRead(joyInButton) == HIGH && isHeld == 0) {
     //Button Released?
     if (debug_rawButton == 1) {
     Serial.println("Button Clicked with no hold.");
     }
     debounceState = 0;
     buttonClicked();
    }
     if (digitalRead(joyInButton) == HIGH && isHeld == 1) {
     //Button Released?
     if (debug_rawButton == 1) {
     Serial.print("Button Released after holding for: ");
     Serial.print(millis() - lastDebounce);
     Serial.println(" mS.");
     }
     debounceState = 0;
     isHeld = 0;
     buttonReleased(millis() - lastDebounce);
   }
   if (digitalRead(joyInButton) == LOW) {
   //Button still pressed??
   isHeld = 1;
   if (debug_rawButton == 1) {
   Serial.print("Button held for: ");
   Serial.println(millis() - lastDebounce);
   }
  }
 }
}
if (debounceState == 0) {
 if (digitalRead(joyInButton) == LOW) {
 //Button pressed?
 debounceState = 1;
 lastDebounce = millis();
 }
}
} //Loop end
//------------------------------------------------------------------------------------------------------------
/*
The sections below all contain an area that looks like this:
//Actions to perform when this is triggered

//End actions to perform
 Feel free to add any actions in between those lines, that you would like to perform when that particular event
is triggered.
*/
//------------------------------------------------------------------------------------------------------------
void buttonClicked() {
 //This is called when the button is clicked without holding.
 if (debug_processedButton == 1) {
   Serial.println("Button Click event fired.");
 }
 //Actions to perform when this is triggered

 //End actions to perform
}
void buttonReleased(int lengthHeld) {
 //This is called when the button is released after holding. lengthHeld contains duration in mS.
 if (debug_processedButton == 1) {
   Serial.print("Button Held event fired, length of hold was: ");
   Serial.print(lengthHeld);
   Serial.println(" Milliseconds.");
 }
 //Actions to perform when this is triggered

 //End actions to perform
}
bool buttonHeld() {
 //This is called to check if the button is still being held. True or 1 if its held, False or 0 if not.
 if (isHeld == 1) {
   return 1;
 }
 else {
   return 0;
 }
 //Actions to perform when this is triggered

 //End actions to perform
}
void joyUp(int moveAmt) { //Called when the joystick is moved "upwards", with the value stored in moveAmt
   moveAmt = constrain(moveAmt, 0, 495);
 if (debug_processedJoystick == 1) {
   Serial.print("Joystick moved Up.");
   Serial.println(moveAmt);
 }
 //Actions to perform when this is triggered

 //End actions to perform
}
void joyDown(int moveAmt) { //Called when the joystick is moved "downwards", with the value stored in moveAmt
   moveAmt = constrain(moveAmt, 0, 495);
 if (debug_processedJoystick == 1) {
   Serial.print("Joystick moved Down.");
   Serial.println(moveAmt);
 }
 //Actions to perform when this is triggered

 //End actions to perform
}
void joyLeft(int moveAmt) { //Called when the joystick is moved "left", with the value stored in moveAmt
   moveAmt = constrain(moveAmt, 0, 495);
 if (debug_processedJoystick == 1) {
   Serial.print("Joystick moved Left.");
   Serial.println(moveAmt);
 }
 //Actions to perform when this is triggered

 //End actions to perform
}
void joyRight(int moveAmt) { //Called when the joystick is moved "right", with the value stored in moveAmt
   moveAmt = constrain(moveAmt, 0, 495);
 if (debug_processedJoystick == 1) {
   Serial.print("Joystick moved Right.");
   Serial.println(moveAmt);
 }
 //Actions to perform when this is triggered

 //End actions to perform
}
//----End Code (copy to here)----
```

Expected Output:

The code provides output to debug the module, as well as the ability to be easily modified to perform any actions the user wants based on input.

See image (right) for example.



```
Button Click event fired.
Button Held event fired, length of hold was:
Joystick moved Up.5
Joystick moved Up.6
Joystick moved Up.7
Joystick moved Up.7
Joystick moved Up.159
Joystick moved Up.184
Joystick moved Up.184
Joystick moved Up.158
Joystick moved Down.5
Joystick moved Down.7
Joystick moved Down.8
Joystick moved Down.31
Joystick moved Down.282
Joystick moved Down.355
Joystick moved Down.248
Joystick moved Left.5
Joystick moved Left.11
Joystick moved Left.19
Joystick moved Left.155
Joystick moved Left.398
Joystick moved Left.401
Joystick moved Left.280
Joystick moved Right.6
Joystick moved Right.19
Joystick moved Right.39
Joystick moved Right.371
Joystick moved Right.495
Joystick moved Right.487
```

This module is useful for…

Some ideas for projects this module is useful in might include:

1. Robotics
2. RC Vehicles
3. Games/Controllers

E-Mail the author by CLICKING HERE.

Need more modules, components or project ideas or kits? Aus Electronics Direct have the best prices, and a fantastic range! (And other products too!)