Willkommen!

Und herzlichen Dank für den Kauf unseres PIR Bewegungssensor Modul! Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Schritte von der Einrichtung bis zur Ausgabe der Werte. Viel Spaß!

Die Funktionsweise unseres PIR-Bewegungsmeldemoduls auf Platine mit Schaltkontakt ist schnell erklärt. Es handelt sich um einen pyroelektrischer Sensors welcher Infrarotstrahlung wahrnehmen kann. Ähnliche Sensoren sind Beispielsweise in kontaklosen Infrarotthermometern verbaut. Über dem Modul ist eine kleine Plastiksphere angebracht, welche die vom Sensor aufgenommene Infrarotstrahlung streut. Dadurch wird der Bereich in dem Bewegung erkannt wird vergrößert, bzw. gelenkt.
Auf der Platine befinden sich zwei Potetiometer zur Ansteuerung, sowie eine Jumperbrücke. Die beiden Potetiometer sind zum einstellen der Empfindlichkeit und der Schaltdauer des im Modul integrierten Timers, die Jumperbrücke zur Konfiguration des gewünschten Ausgabeformates (single Trigger, repeat Trigger).

Die wichtigsten Informationen in Kürze

» **Abmessungen:** 33mm x 25mm x 25mm

» **Verbindung:**

3.3 - 5 V	VCC
GND	Masse
S	DigitalOut

» **Temperaturbereich:** -65 - 150 °C
» Programmierung über Digital Pin

Auf den nächsten Seiten findest du Informationen zur
» ***Einrichtung der Hardware***
und eine Anleitung für
» ***das Auslesen der Sensordaten***.

Diese Anleitung setzt voraus, dass du weißt, wie du Sketche auf einen Arduino hochlädst und den Serial Monitor verwendest! Nähere Informationen finden Sie in unserem E-Book zum Arduino.

Diese Anleitung setzt weiter voraus, dass du weißt wie man einen Raspberry in Betrieb nimmt und die GPIOs nutzt. Nähere Informationen findest du im E-Book für den RaspberryPi

Alle Links im Überblick

Bibliotheken:

» Bibliothek (Rpi): https://github.com/WiringPi/WiringPi

Programmieroberflächen:

» Arduino IDE: *https://www.arduino.cc/en/Main/Software*

» Web-Editor: *https://create.arduino.cc/editor*

» Arduino-Erweiterung für SublimeText: *https://github.com/Robot-Will/Stino*

» Arduino-Erweiterung "Visual Micro" für Atmel Studio oder Microsoft Visual Studio: *http://www.visualmicro.com/page/Arduino-for-Atmel-Studio.aspx*

Arduino Tutorials, Beispiele, Referenz, Community:

» *https://www.arduino.cc/en/Tutorial/HomePage*

» *https://www.arduino.cc/en/Reference/HomePage*

Interessantes von AZ-Delivery

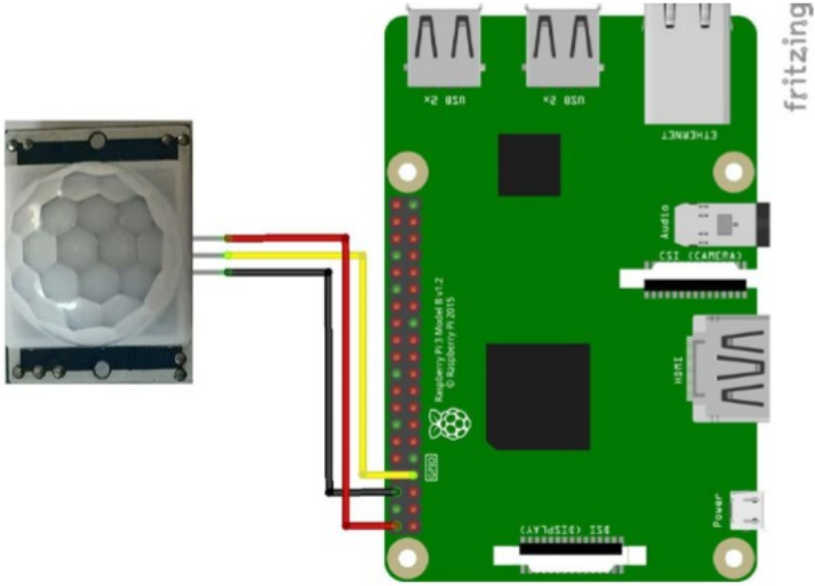
» Arduino Zubehör: *https://az-delivery.de/collections/arduino-zubehor*

» AZ-Delivery G+Community: *https://plus.google.com/communities/115110265322509467732*

» AZ-Delivery auf Facebook: *https://www.facebook.com/AZDeliveryShop/*

Einrichtung der Hardware

Dank dem verbauten Schaltkontakt mit integriertem Timer ist dieser Sensor sehr controllerfreundlich, er kann mit 3.3 – 5V betrieben werden. Auf dem Schaltmodul kann man mithilfe des Potentiometers den Schwellenwert für den Schaltkontakt sowie die Schaltzeit einstellen. Der Signal Pin liefert ein High Signal, den Intervall hierzu können Sie über die Jumperbrücke einstellen. Steht der Jumper an der Standartposition (an der Platine aussen) wird der Pin geschalten und fällt nach Ablauf des Timers auf Low zurück. Ist die Brücke zwischen Pin 2 und 3 arbeitet Ihr Sensor im repeat Mode und zeigt kontinuierlich Veränderungen, je nach Bewegung. Jeder Digitale Pin am Arduino kann verwendet werden.



Beispielcode am RaspberryPI:

Zum Auslesen des GPIO verwenden wir die GPIO Bibliothek wiringPi.

sudo apt-get -y install git-core
cd ~
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build

Nach der Installation von wiringPi können wir das gleich testen:
gpio -v

gpio readall

```
pi@raspberrypi3p:~/wiringPi $ gpio readall
 +-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
```

Das Auslesen der Sensordaten

cd ~
touch pir.py
nano pir.py

import RPi.GPIO as GPIO import time

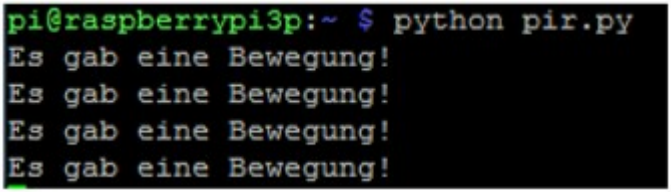
SENSOR_PIN = 4

GPIO.setmode(GPIO.BCM) GPIO.setup(SENSOR_PIN, GPIO.IN)

def bewegung(channel): print('Es gab eine Bewegung!')

try: GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING, callback=bewegung) while True: time.sleep(100) except KeyboardInterrupt: print "Programm wurde beendet." GPIO.cleanup()

python pir.py



Und weitergehts mit der Abfrage der Daten an einem Arduino
Zum Anschluss können Sie jeden beliebigen Digital-Pin nutzen, in unserem Beispiel verwenden wir Pin 3

Wir verbinden VCC mit 5V, GND mit GND und S mit D3 am Uno

der BeispielCode sieht so aus:

```
int PIR=3;
int led=13;

void setup() {
pinMode(PIR, INPUT);
pinMode(led, OUTPUT);
Serial.begin(9600);
}

void loop() {
if(digitalRead(5) == HIGH)
{
 digitalWrite(13, HIGH);
 Serial.println("motion detected");

}
else {
 digitalWrite(13, LOW); // setting led to low
 Serial.println("scanning");
}
}
```

Den ausgegebenen Digital-Wert können wir am Seriellen Monitor oder über Pin 13 (BuildInLED)

Du hast es geschafft! Herzlichen Glückwunsch!

Ab jetzt heißt es lernen und ausprobieren. Du weißt nun, wie du die Feuchte von Substrat bestimmen kannst Jetzt kannst du versuchen, die Werte praktisch einzusetzen – vielleicht zum Ein- und Ausschalten einer automatischen Bewässerung? Diesen und noch mehr Hardware findest du natürlich in deinem Online-Shop auf:

https://az-delivery.de

Viel Spaß!

Impressum

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