Alarm System

on the WemosD1Mini ESP8266 Module

Introduction

My friends and i built a hut in the woods, wheres nobody around and no Electricity available. Since we had a burglar, we decided to built an alarm system. It should be easy to build, not expensive and needs a low energy consumption since there's no electricity around.
So we took our programming skills to make a Telegram bot, which sends messages to any user connected to it and a device to collect sensor data. As interface between the Telegram bot and the ESP module we're using a free account at thingspeak.com.

How does it work?

Well first there was just the board and some hardware. For the existing hardware it was'nt easy to reset the ESP8266 when the door was opened so we needed an additional door sliding contact, you know those doorbell switches some old stores use for the bell to ring when someone opens the door. So with such a switch added from ground to the reset pin the functionality was complete and here is how it works.
The Program does an automated sensor data update every 45 seconds so we always know the battery voltage and if there's some movement even without triggering the door. When the Door is triggered, the ESP8266 starts its loop from the beginning. It starts by the setup, collects sensor data, processes the data and decides if the MOS-module has to be turned on or off. After that it proceeds to connect to the WiFi and uploads gathered Data to the Thingspeak cloud.
The MOS-module will be turned on as long as there's movement detected or the door/window is open. When the door/window is closed and no movement detected the MOS stays on for five more minutes.

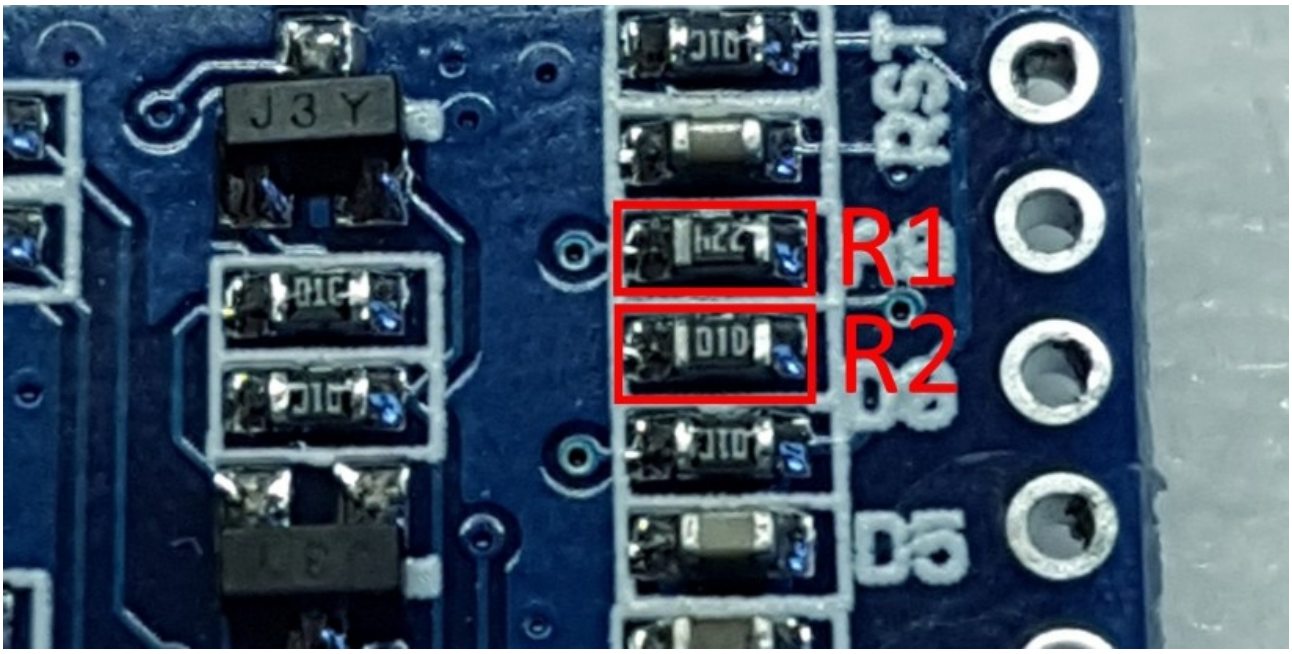
Hardware

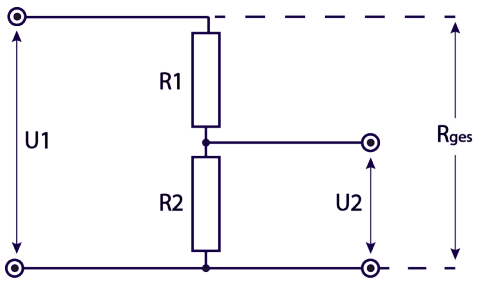
So we took the cheapest and most suitable Hardware we got laying around and started. For this Part it was:

Amount	Part
1	Wemos D1 Mini Development board
1	Reed switch (from old Coffee machine)
1	PIR Motion Sensor Module
1	DHT22 Temperature and Humidity Sensor
1	MOS Module
1	Jumper Wires
1	Door sliding contact
2	Step-Down Buck converter
1	Mobile LTE-Router WiFi Access point
1	20Watt Peak Solar Panel
1	Old USB Hub
2	SMD 0603 Resistors 1x 10MΩ, 1x 220KΩ
X	Wires, connectors and stuff

Modifications of the Voltage Divider

On the Wemos D1 Mini Module i had to do a small modification for the Module to be able to measure higher Voltages … so i made it capable of measuring Voltages up to 48 Volt! Thanks to a tutorial on nerddiy.de i found out which ones are the Voltage divider Resistors. Then i calculated suitable values for replacement resistors and found a fit with R1 = 10MΩ and R2 = 220KΩ . You'll might notice there's already a 220KΩ resistor on that board! Yeah that's right. Just use the top one and solder it one place underneath. In the now free spot place the 10MΩ resistor.





Object	Value	Comment
U1	0-48 Volt	Maximum Input Voltage for example with multiple batteries
U2	0-1 Volt	Maximum Input Voltage the ESP8266 ADC input is capable of
R1	10MΩ	Calculated Value to reach the max. 1V
R2	220KΩ	Calculated Value to reach the max. 1V
Rges	10,2MΩ	Sum of the Resistors

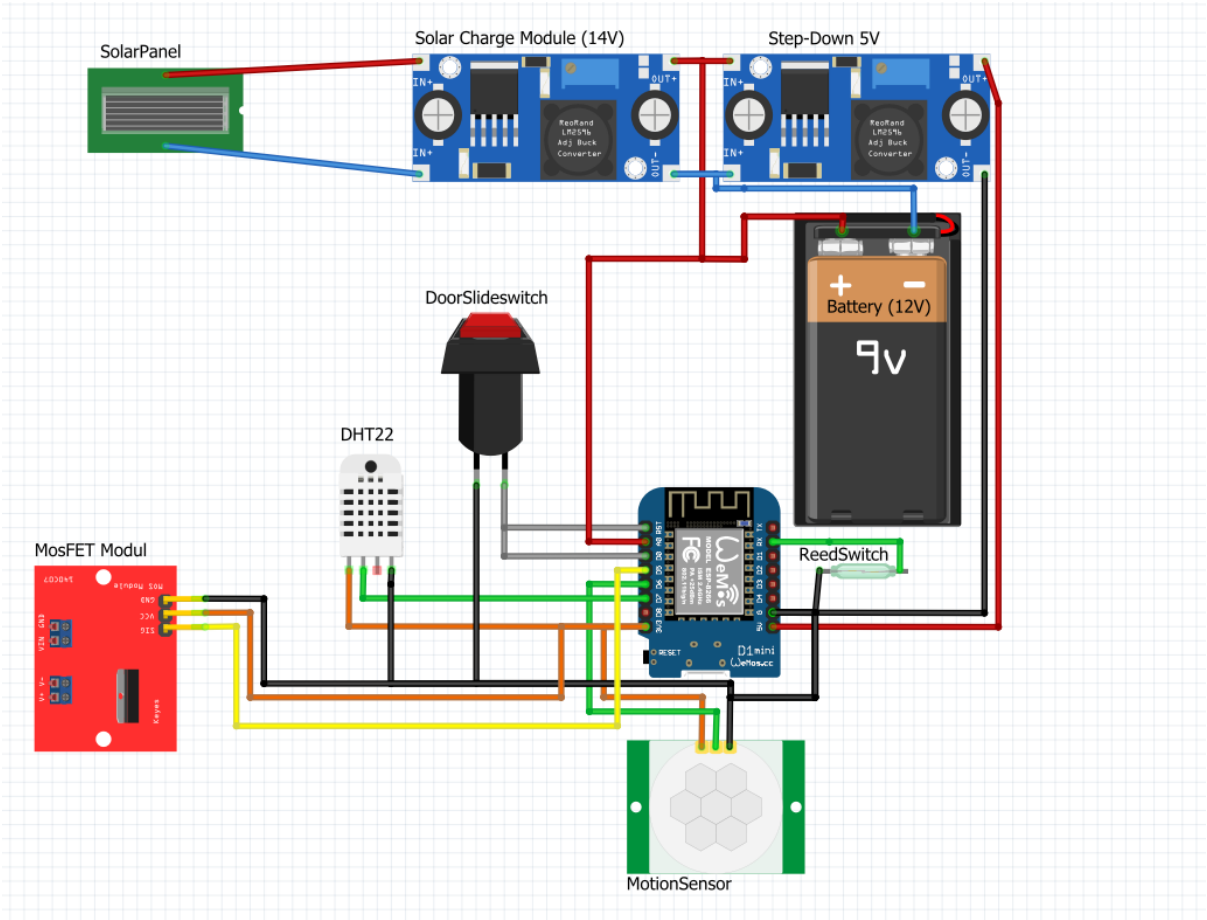
Calculations done with:
https://www.peacesoftware.de/einigewerte/spannungsteiler.html

Wiring

The wiring should've been the easiest part … well but the ESP did not play along.
According to a table i found Online (HERE), the ESP8266 has more specific needs for it's IO pins. So with this guide i edited the program code for the ESP8266 to play along.

Label	GPIO	Input	Output	Notes
D0	**GPIO16**	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	**GPIO5**	OK	OK	often used as SCL (I2C)
D2	**GPIO4**	OK	OK	often used as SDA (I2C)
D3	**GPIO0**	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	**GPIO2**	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	**GPIO14**	OK	OK	SPI (SCLK)
D6	**GPIO12**	OK	OK	SPI (MISO)
D7	**GPIO13**	OK	OK	SPI (MOSI)
D8	**GPIO15**	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	**GPIO3**	OK	RX pin	HIGH at boot
TX	**GPIO1**	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	**ADC0**	Analog Input	X	

Here is the final Wiring made with Fritzing.



Code

Coding wasn't that hard but figuring out the whole battery measuring stuff was… However here i explain how you can make some settings in the code to fit your needs.

On the tab „secrets.h“ you can set up your WiFi name and password. Its bound to the code with the function #include "secrets.h"

On the section „Variables and Connections“ you can set the sleeptime and the time that the FET should be on. With int fettime = 10; its possible to set the time in minutes for the FET to be on. Please notice that the number has to be double the minutes you want the FET to be turned on! int loopcount = 0; is only a variable where the microcontroller stores the amount of loops already passed. If there's movement or the door/window is open the loopcount will be set to 0 again, which extends the FET time. Last but not least we've the entry int sleeptime = 45; where the time for deepsleep is set in seconds.

In the next line you'll find float multiplicator = 0.0437;. This variable is responsible for calculating the battery voltage. You can calculate your own multiplicator by using the serial monitor. It gives you a reading of the vread value. Just apply a voltage of your choice (1-48 Volt after you've done the modification on the Voltage divider) for example 10Volts and the vread will give you a reading of something around 231. Next you just need to divide the Voltage with the reading. So dividing 10 by 231 equals 0,4329. After editing this part of the Code it will generate a Voltage value, which accuracy you can fine-tune by simply increasing or decreasing the multiplicator Value a little bit.

The last part of the Variables and Connections section is for the connections from and to the Wemos D1 mini. I would suggest to leave this part as it is since not all of the GPIO Pins are capable to be used as in or output, as you can see from the table at Wiring.