Using ESP8266 to monitor voltage

… but could just as well monitor anything you can find a sensor for

Why did I want to do this?

- PROBLEM: One of our cars is not used often and the battery was run down by a reading light accidently left on
- SOLUTION: Construct a monitor for the battery voltage to notify me if the battery voltage is lower than a preset value

How?

- Programmed an ESP8266 to check the battery voltage once an hour and publish a message via MQTT once a day if all is good or immediately if a low voltage or some error is detected
- Used an MQTT server I already have running on a Raspberry Pi Zero W
- Wrote a small program to run continuously on my Pi to receive the messages and just log them
- Run another small program once a day to check that log and notify me if there is a problem

Hardware Used

- HiLetgo Wemos D1 Mini Development Board ESP8285 V1.0.0 1MB Flash Lite Wireless WiFi Internet Development Board Wemos D1 Mini ESP8285 from Amazon … 2 for $11
- Voltage regulator to convert the 12v car battery supply to the 5v needed by the ESP
- Two resistors to form a voltage divider circuit since the ESP can handle a maximum of 3.3 volts on the A0 analog-to-digital pin

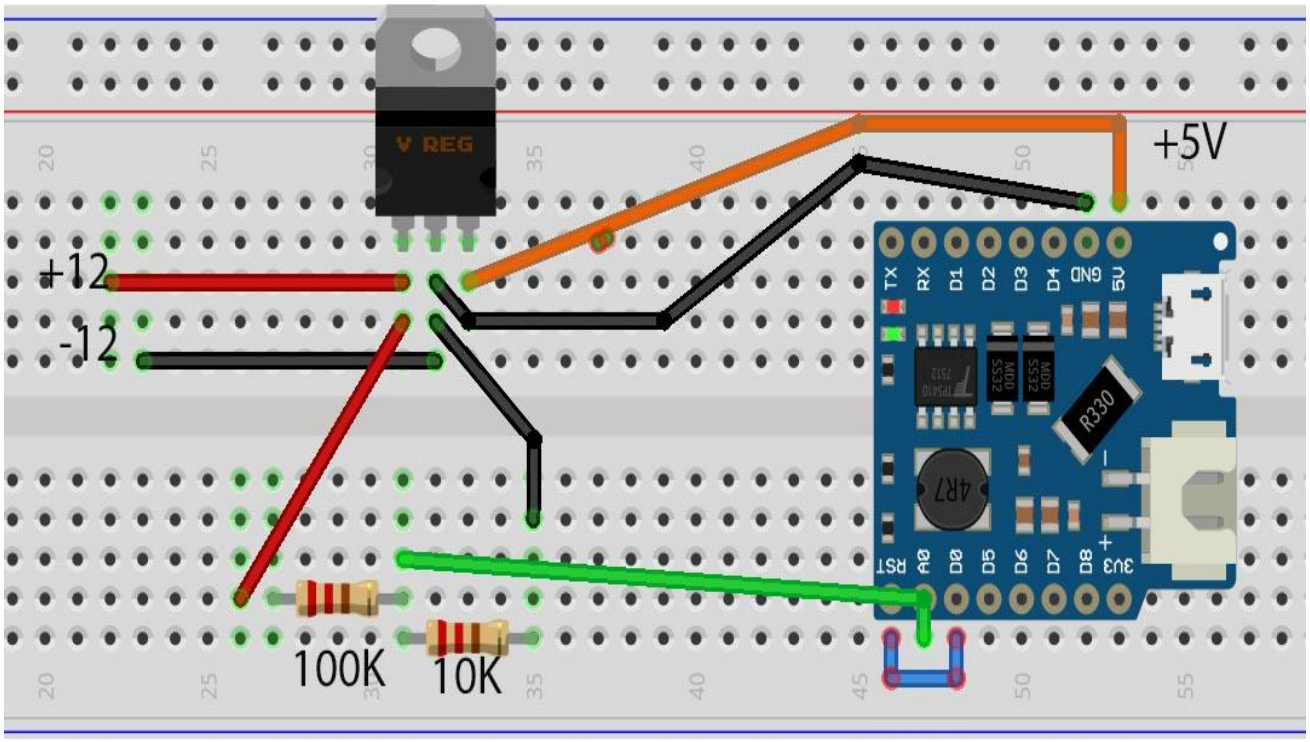
Wemos D1 Mini ESP8266

Some details

- Analog pin A0 supports up to 3.3 volts which it converts to a digital value in the range 0-1024
- In order to use the deep sleep mode provided by the ESP8266 there must be a jumper connecting pins RST and D0
- Maximum time for deep sleep is 2^{32} microseconds or about 71 minutes
- In order to write a new program to the ESP, the RST-D0 jumper must be removed !!!

Picture of device

Circuit breadboard



ESP8266 Program operation

- The setup() function runs one time when power is applied to the ESP; the loop() function then runs continuously
- Waking up from deep sleep is the same as the initial power up – the setup() function runs, etc.
- In order to save information over resets or wakeups from deep sleep, the ESP provides EEPROM storage. This is one of two ways to save data over deep sleep cycles. (SPIFFS is the other.)

EEPROM usage

- Electrically Eraseable Programmable Read-Only Memory
- #include <EEPROM.h>
- EEPROM.begin(size) where size is 4-4096
- EEPROM.put(offset,variable-name)
- EEPROM.get(offset,variable-name)
- EEPROM.commit() actually writes data back to the EEPROM memory

Before we look at the ESP code …

- sleep() enters deep sleep for 1 hr
- getVoltage() reads the voltage
- setup_wifi() tries to connect to wifi
- connectMQ() tries to connect to MQTT
- setup() main program
- loop() required but never used