The Sensor body

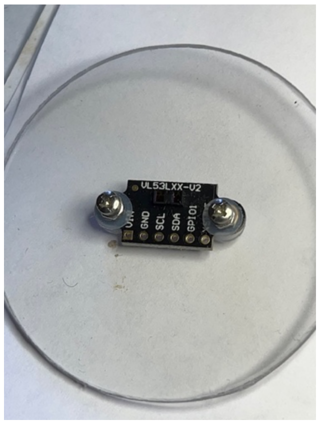
The sensor was prepared with 4 x 28 awg, 6-inch wires that will connected later to the ESP32 board. The sensor body was made from a 2in PVC coupling, threaded on one end for the tank, and “slip” fitting on the other. A sheet of clear 1/16in Lexan was cut with a 2 ½ in hole saw to make the mounting plate for the sensor. (The inside diameter of the 2 1/2in hole saw is just the correct size for the mounting plate) The sensor was centered in the Lexan and mounted to the Lexan with 2 x 2.5mm stainless screws, thru the Lexan, with a nut that was pre-tighten, and a nylon washer to get the correct distant from the sensor to the Lexan, then the sensor, another nylon washer on top and finally a nut. Lexan was selected as its resistant to fuel-oil.

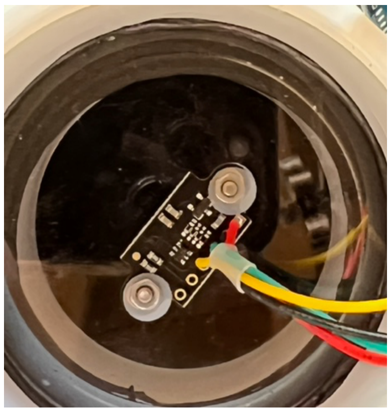
The PVC coupling was prepared with a bead of Permatex 82180 oil resistant silicon on the inside ridge of the coupling. The sensor assembled was then pressed inside the coupling onto the black silicon to form a vapor free seal from the tank to the sensor. A weight was added to keep pressure on the bond, and it was left to cure for 12hr.

Above the sensor, a thin piece of cardboard was cut with hole in the center to route the cable to the ESP32. This allowed the ESP32 to free float above the sensor without any issue of contact between the sensor and the ESP32 board.

The sensor wires were connected to the ESP32 D1 style board to GND, +3.3V, SDA on pin xx and SCL to pin xx. To supply power, a cable was attached to GND and to VCC pins on the ESP32 board. This cable was routed to a cable gland mounted on top of a 2in PVC plug. The cable was connected to a 5V power source. (NOTE: 5V Only!!)

Programming of the ESP32 was done via its micro-USB connector.





Programing the firmware.

The sensor use on the project is an ST-Micro VL53L1X time of flight optical sensor, using the Tasmota Open source IOT firmware for ESP32 with its Berry scripting language. It provides a rich framework for develop IOT projects like this. We looks at other sensor options like SR04M ultrasonic sensor, but decided on the VL53L1X optical device. Another option would be the older VL53L0X with some minor changes to the software.

Tasmota has many, many options, it support a web interface and a full MQTT delivery of sensors data to downstream device like Node-Red, Grafana or home automation system, it also has a very rich scripting language (Berry) available.

The standard release (Ver: 10.1.0.0) that we used did not load the driver for the VL53L1X as a standard option, and we then needed to re-compile the program, that was done via Visual Studio Code and a copy of the source-tree from GitHub. A pre-compile binary is available in GitHub for this project.

It appears that the I2C driver for the VL53L1X does-not play well with some of the other pre-define drivers in Tasmota. To solve this issue we needed to disable all of the I2C driver except this one…

All change were made in one file: tasmota/user_config_override.h
and re-compile of Tasmota ESP32.

```
This will enable I2C on GPIO 22 and 23

#define USER_TEMPLATE "{\"NAME\":\"Tank Sensor
VL53L1X\",\"GPIO\":[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,608,640,0,1,1,1,0,0,0,0,1,1,1,
1,1,0,0,1],\"FLAG\":0,\"BASE\":1,\"CMND\":\"SetOption8 1\",\"CMND\":\"Module 0\"}"

    #ifndef USE_VL53L1X
    #define USE_VL53L1X
    #endif

    #define USE_BERRY_DEBUG

    #define I2CDRIVERS_0_31  0x00000000
    #define I2CDRIVERS_32_63 0x00400000  // enable only device VL53L1
    #define I2CDRIVERS_64_95 0x00000000
```

To load this file, compile Tasmota32 with the option above...
Then Load the new binary image in your ESP32 and re-boot it.
Open the web page for this device, select Console, then Manage File System
Rename this Berry file to "autoexec.be", then upload it to the ESP32 file system.
Reboot Tasmota, this Berry file will run after re-booting.