**07/03/2022 – Sunday - 220703-STM32F4\_Using\_CubeMX\_and\_IAR\_STM32F4-Discovery-ESC**

The object of this project is to make a Video to document the previous work done on the STM32F4-Discovery-ESC system.

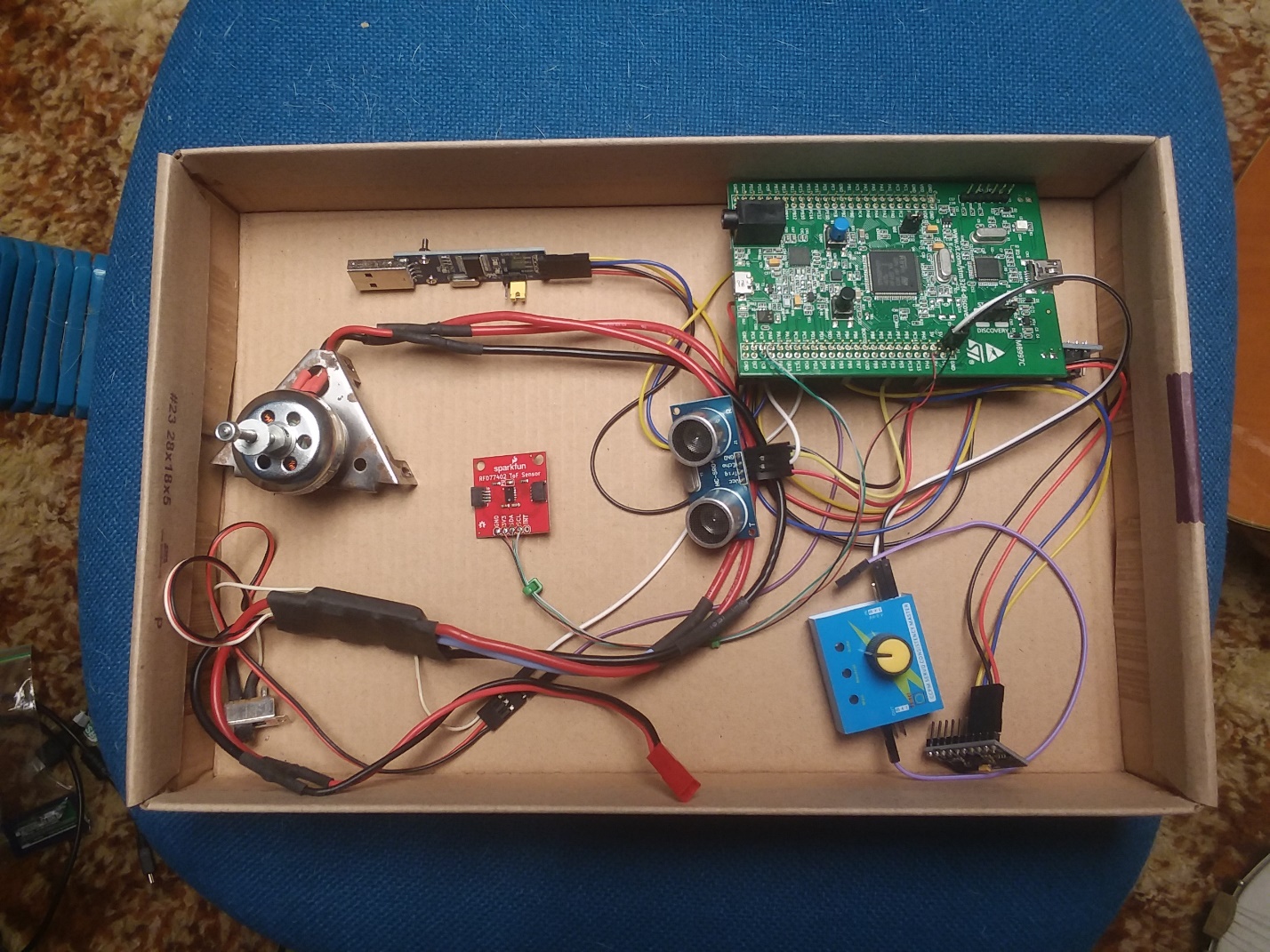
**Tuesday – 09/10/2019**

**STM32F4-Discovery-ESC**

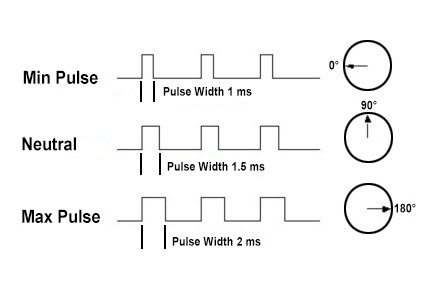
This was originally hosted on the “embed01” PC, and then transferred to the “embed02” PC, and this write-up is based on “example009”. This is a STM32F4-Discovery Board. A 7.4 Vdc battery is used. This was also put on the “seminar01” PC.

It contains:

1. stm32f4-discovery board
2. Brushless Motor
3. Black ESC
4. Servo Simulator – The pulse width in seconds is read. (1 to 2 msec = normal.)
5. PL2303 – Serial UART to USB Converter – Used for instrumentation.
6. HC-SR04 – Sonar
7. RFD77402 – ToF Sensor
8. MPU-9250 – IMU
9. AT24C256 – I2C EEPROM 32,768x8 = 256Kbits



“20190910\_095633.jpg”



A servo signal has a 1 to 2 msec range.

**Connections:**

**MPU-9250 to STM32F4-DISCOVERY**

|  |  |  |
| --- | --- | --- |
| **MPU-9250** | **STM32F4-DISCOVERY** | **Color** |
| VCC | 3V | RED |
| GND | GND | BLK |
| SCL | PB6 – I2C1\_SCL | BLU |
| SDA | PB7 – I2C1\_SDA | YEL |

The I2C1 is running at 400,000 Hz.

**AT24C256 to STM32F4-DISCOVERY**

|  |  |  |
| --- | --- | --- |
| **AT24C256** | **STM32F4-DISCOVERY** | **Color** |
| VCC | 3V | RED |
| GND | GND | YEL |
| SCL | PB10 – I2C2\_SCL | BLK |
| SDA | PB11 – I2C2\_SDA | BLU |

The I2C2 is running at 400,000 Hz. – This is only used in start-up in this application.

**RFD77402 to STM32F4-DISCOVERY**

|  |  |  |
| --- | --- | --- |
| **RFD77402** | **STM32F4-DISCOVERY** | **Color** |
| 3V3 | 3V | RED |
| GND | GND | BLK |
| SCL | PC9 – I2C3\_SCL | WHT |
| SDA | PA8 – I2C3\_SDA | GRN |

The I2C3 is running at 400,000 Hz.

**HC-SR04 to STM32F4-DISCOVERY**

|  |  |  |
| --- | --- | --- |
| **HC-SR04** | **STM32F4-DISCOVERY** | **Color** |
| VCC | 5V | RED |
| GND | GND | BLK |
| Trig | PB13 – GPIO\_OUTPUT | WHT |
| Echo | PE5 – TIM9\_CH1 | YEL |

Channel 1 = Input Capture Direct Mode

Channel 2 = Input Capture Indirect Mode

**Servo Simulator to STM32F4-DISCOVERY**

|  |  |  |
| --- | --- | --- |
| **Servo Simulator** | **STM32F4-DISCOVERY** | **Color** |
| IN + | 5V | WHT |
| IN - | GND | BLK |
| OUT S | PB14 – TIM12\_CH1 | VIO |

Channel 1 = Input Capture Direct Mode

Channel 2 = Input Capture Indirect Mode

**ESC to STM32F4-DISCOVERY**

|  |  |  |
| --- | --- | --- |
| **ESC** | **STM32F4-DISCOVERY** | **Color** |
| GND | GND | BLK - VIO |
| S | PD15 – TIM4\_CH4 – d\_pwm\_fix | WHT |

PWM Generation CH4

**PL2303 to STM32F4-DISCOVERY**

|  |  |  |
| --- | --- | --- |
| **PL2303** | **STM32F4-DISCOVERY** | **Color** |
| VCCIO | 5V | RED |
| GND | GND | BLK |
| TXD | PD9 – USART3\_RX | BLU |
| RXD | PD8 – USART3\_TX | YEL |

UART set to 115200-8-N-1

**Update Rate:**

Timer 3 is used to set the 500 Hz update rate. The PSC is set to 31 and the Auto Reload Register (ARR) is set to 5249. The timer runs at 84,000,000 Hz. The equation is 500 = (84,000,000)/(PSC+1)/(ARR+1)

**Getting the Motor to Run:**

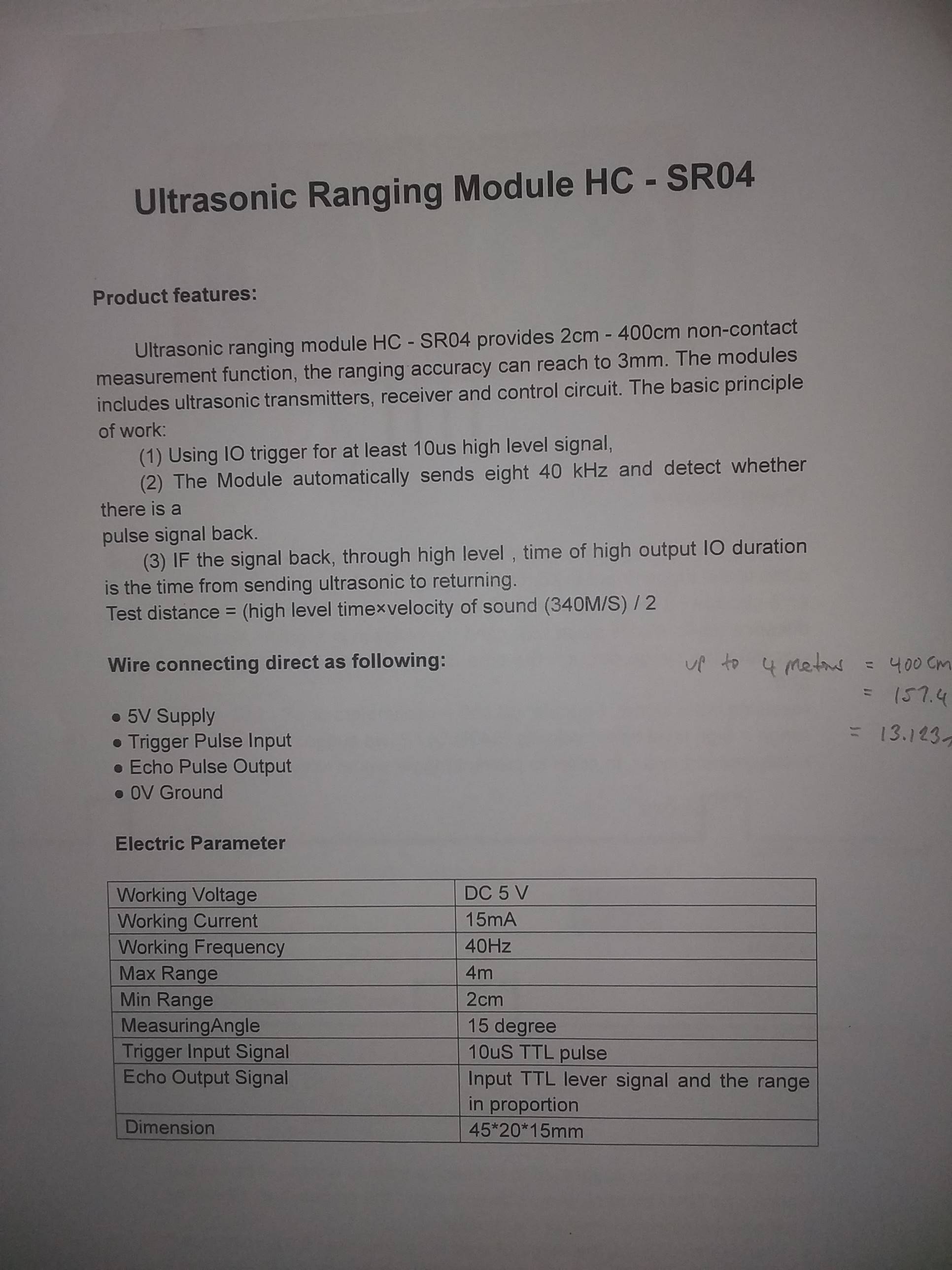
The 7.4 Vdc battery needs to be connected to the ESC to run the motor. In the debugger, the default for “d\_pwm\_fix” is set to 1000. This needs to be changed to 1100 to start the motor running. If set to 1150, the motor will run faster. Setting it back to 1000, will stop the motor.

**AT24C256 Test:**

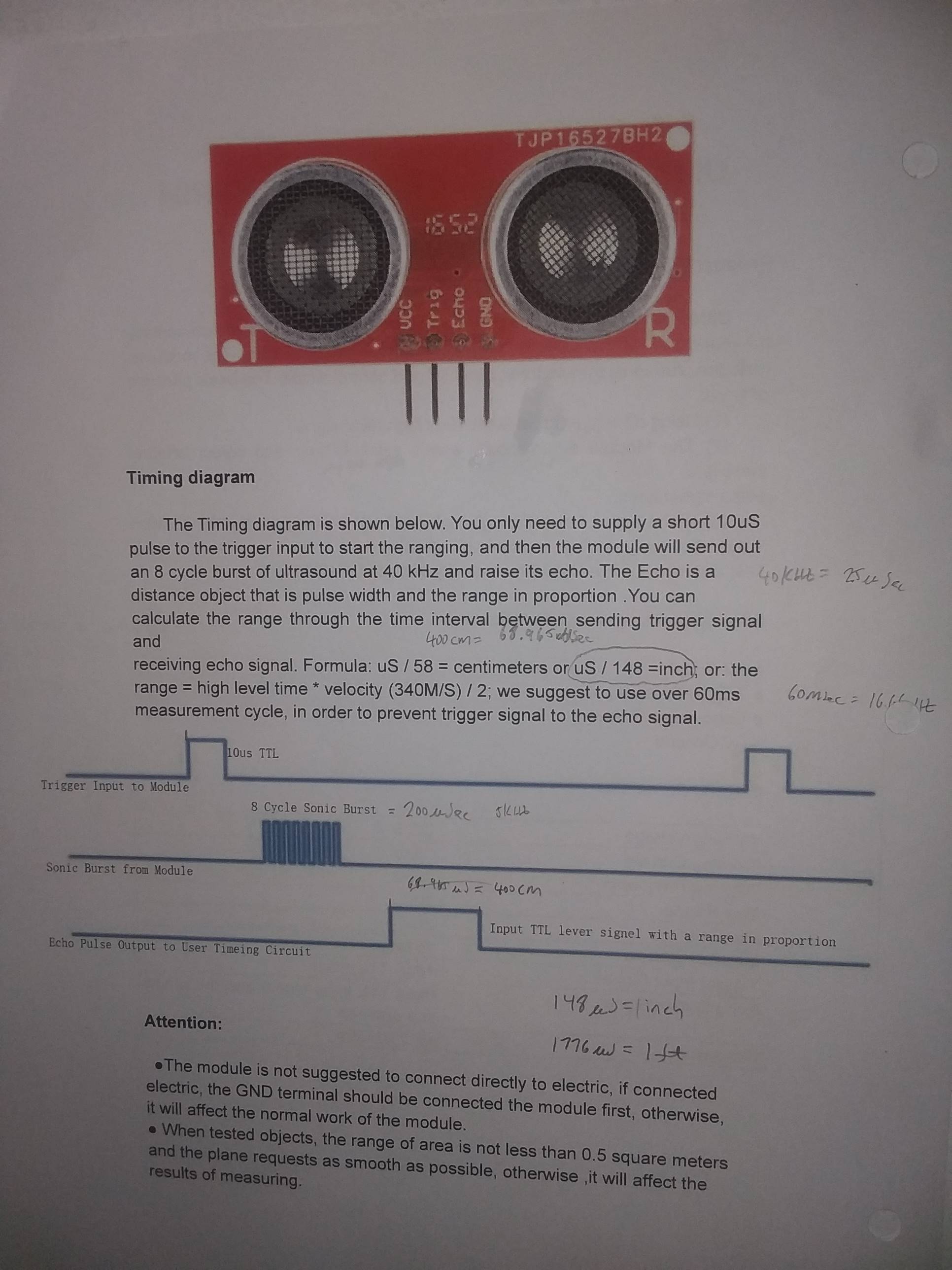
The parameters “AT24C256\_M0000”, “AT24C256\_M0001”, “AT24C256\_M0002”, & “AT24C256\_M0003” are written into at startup and then read using the I2C2 interface. These parameters are examined in the debugger to prove they were read.

**HC-SR04 Test:**

Timer 9 is used to measure the pulse generated by the sonar. It was found that the sonar works well. But the sound from the brushless motor interferes with it. The parameter “sonar\_pulse\_echo\_inches” is used to read the distance in inches. It is limited to 157.48 inches which is 4 meters. PB13 is used to trigger the process. Every 25 frames are skipped to give the sonar a 20 Hz update rate. 148 micro-seconds = 1 inch.



“20190910\_194717.jpg”



“20190910\_194738.jpg”

The PSC is set to 127 and the ARR is set to 65535. So the timer counts at 168,000,000/(128) = 1,312,500 Hz which is 0.761 micro-seconds/count. The following method is used to compute the distance:

sonar\_pulse\_echo\_capture1 = HAL\_TIM\_ReadCapturedValue(&htim9, TIM\_CHANNEL\_1);

sonar\_pulse\_echo\_capture2 = HAL\_TIM\_ReadCapturedValue(&htim9, TIM\_CHANNEL\_2);

sonar\_pulse\_echo\_captured = sonar\_pulse\_echo\_capture2 - sonar\_pulse\_echo\_capture1;

sonar\_pulse\_echo\_sec = 128.0/168000000.0\*(float32\_t)sonar\_pulse\_echo\_captured;

sonar\_pulse\_echo\_inches = sonar\_pulse\_echo\_sec/0.000148;

**Wednesday - 09/11/2019**

**STM32F4-Discovery-ESC (Cont.)**

**RFD77402 Test:**

This uses the I2C3 interface. It can read the distance to your hand. It has the I2C address of 0x4C. It has a 0.1 to 2 meter range. The parameter “RFD77402\_Height\_Inches” is read using the debugger.

**Servo Simulator Test:**

Timer 12 is used to read the pulse width in seconds. The knob can be turned to increase the pulse width. It has a normal range of 0.001 to 0.002 seconds. The parameter “pulse\_sec” is read using the debugger.

/\* Read the servo simulator.\*/

pulse\_capture1 = HAL\_TIM\_ReadCapturedValue(&htim12, TIM\_CHANNEL\_1);

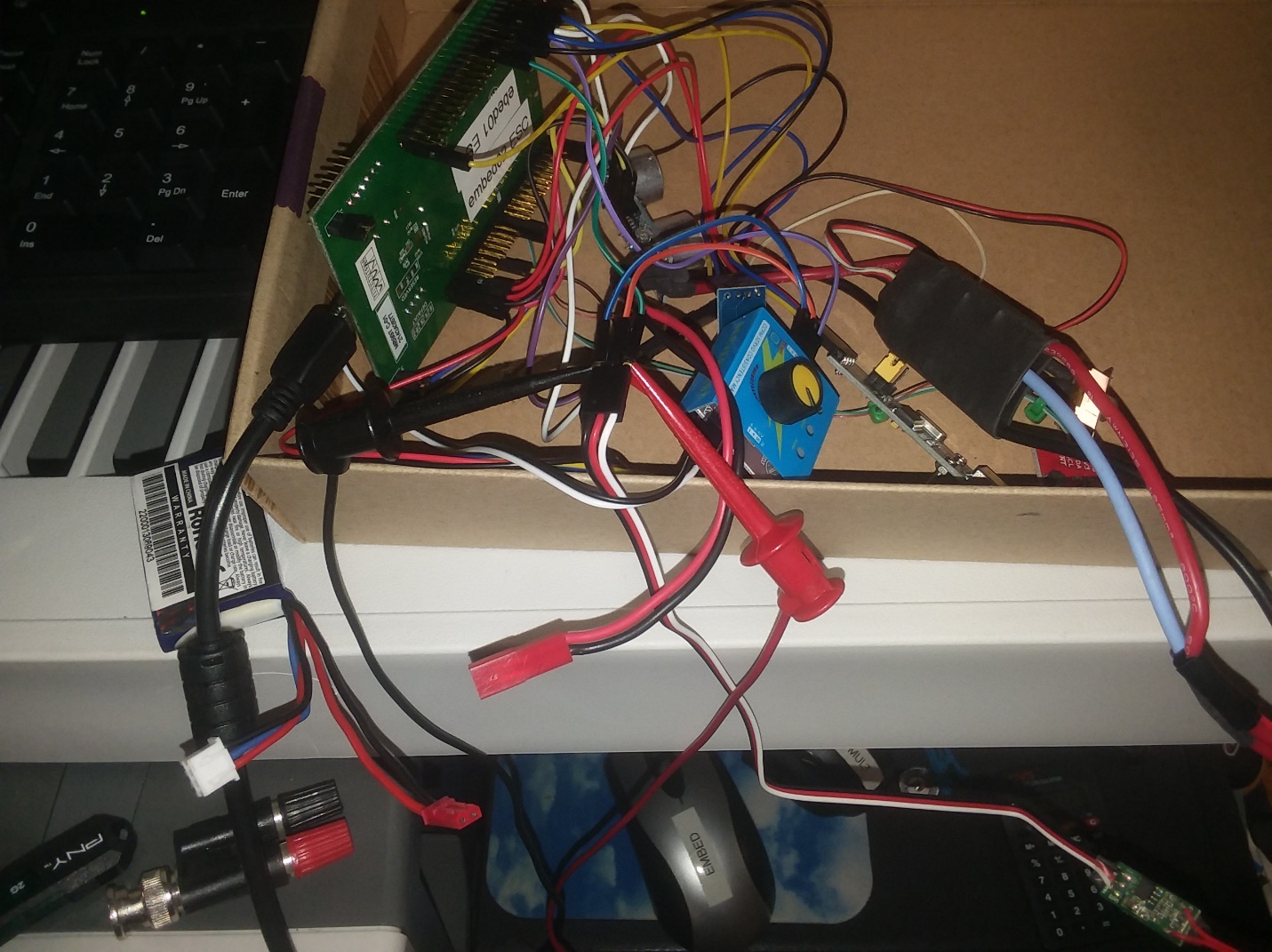
pulse\_capture2 = HAL\_TIM\_ReadCapturedValue(&htim12, TIM\_CHANNEL\_2);

pulse\_captured = pulse\_capture2 - pulse\_capture1;

pulse\_sec = 128.0/84000000.0\*(float32\_t)pulse\_captured;

The scaling is set to 1.523 micro-seconds per count. So 1 msec will be 656 counts.

**STM32F4-Discovery-ESC – example010-ESC on “embed02” – Add RPM Sensor**



“20190914\_191015.jpg”

The RPM Sensor puts out a pulse that is proportional to the RPM. The waveform has a frequency. Taking the frequency in hz and multiplying the value by 10 will give the speed in RPM. However, the signal is read in an asynchronous way. The frame rate of the system is set to 500 Hz. The frequency of the waveform may go up to 2,000 hz which is 20,000 RPM. Timer 1 was set up to read the rising edge and falling edge of the waveform. It was found that the waveform does not have a 50% duty cycle and a fudge factor of 2.3 needs to be added to get the correct frequency rather than 2. Pin 1 of the Red JST connector is connected to the BLK wire of the ESC and Pin 2 of the Red JST connector is connected to the RED wire of the ESC. TIM1\_CH1 is used with is pin PE9. The clock of the timer is 168,000,000 Hz. The PSC is set to 127. Make sure 5Vdc and GND are connected to the Hobbywing RPM Sensor Card. It was observed on an oscilloscope when the pulse was 1.30 ms the period was 3.040 ms. The ratio is 2.33. At a higher speed, the pulse was 0.660 ms and the period was 1.480 ms. So, a fudge factor of 2.3 is needed.

/\*

name: example010-ESC

date: 09-14-2019

purpose:

example010-ESC - Add RPM Sensor Hobbywing - TIM1\_CH1 = PE9.

\*/

/\* TIM1\_CH1 = PE9 - Read RPM Sensor Pulse. \*/

uint32\_t RPM1\_pulse\_capture1 = 0;

uint32\_t RPM1\_pulse\_capture2 = 0;

uint32\_t RPM1\_pulse\_captured = 0;

float32\_t RPM1\_pulse\_sec = 0.0;

float32\_t RPM1 = 0.0;

uint8\_t RPM1\_valid = 0;

/\* Start the Input Capture Timer to read the RPM1 Sensor. \*/

HAL\_TIM\_IC\_Start(&htim1, TIM\_CHANNEL\_1);

HAL\_TIM\_IC\_Start(&htim1, TIM\_CHANNEL\_2);

/\* Read RPM1 sensor.\*/

RPM1\_pulse\_capture1 = HAL\_TIM\_ReadCapturedValue(&htim1, TIM\_CHANNEL\_1);

RPM1\_pulse\_capture2 = HAL\_TIM\_ReadCapturedValue(&htim1, TIM\_CHANNEL\_2);

RPM1\_pulse\_captured = RPM1\_pulse\_capture2 - RPM1\_pulse\_capture1;

RPM1\_pulse\_sec = 128.0/168000000.0\*(float32\_t)RPM1\_pulse\_captured;

if(RPM1\_pulse\_sec > 0.0001 && RPM1\_pulse\_sec < 0.02 )

{

RPM1 = 10.0/RPM1\_pulse\_sec/2.3;

RPM1\_valid = 1;

}

else

{

RPM1 = RPM1;

RPM1\_valid = 0;

}