

# GAMBIT CHALLENGE

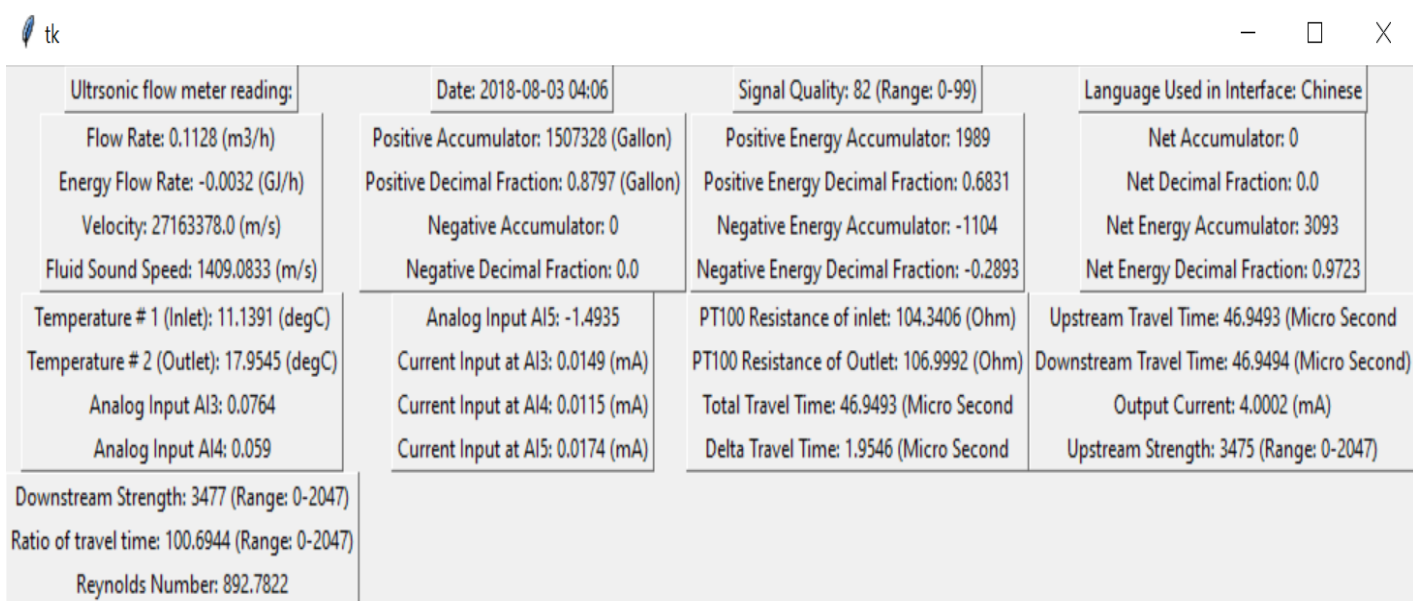
## Task

Parsing the Modbus data

Create a program that parses the data, converts it to human readable data like integers, decimals and strings and presents it in a nice way. Depending on your skills and interests you can create a web service that will provide the conversion data, or you could even create a UI to visualize the data somehow, it is entirely up to you what you make of it!

## Solution

There are two graphical user interface (GUI) created for this task. The first GUI reads Modbus registers values, decodes the respective register values into readable format, and display on GUI. The second GUI allows the user to write register values, encode the respective register values into two bytes list, and store these into respective Modbus registers. However, there will be some discrepancies when writing on GUI due to the limitation of the library used in the application. The first GUI is illustrated in Figure 1.



Ultrasonic flow meter reading:	Date: 2018-08-03 04:06	Signal Quality: 82 (Range: 0-99)	Language Used in Interface: Chinese
Flow Rate: 0.1128 (m3/h)	Positive Accumulator: 1507328 (Gallon)	Positive Energy Accumulator: 1989	Net Accumulator: 0
Energy Flow Rate: -0.0032 (GJ/h)	Positive Decimal Fraction: 0.8797 (Gallon)	Positive Energy Decimal Fraction: 0.6831	Net Decimal Fraction: 0.0
Velocity: 27163378.0 (m/s)	Negative Accumulator: 0	Negative Energy Accumulator: -1104	Net Energy Accumulator: 3093
Fluid Sound Speed: 1409.0833 (m/s)	Negative Decimal Fraction: 0.0	Negative Energy Decimal Fraction: -0.2893	Net Energy Decimal Fraction: 0.9723
Temperature # 1 (Inlet): 11.1391 (degC)	Analog Input AI5: -1.4935	PT100 Resistance of inlet: 104.3406 (Ohm)	Upstream Travel Time: 46.9493 (Micro Second)
Temperature # 2 (Outlet): 17.9545 (degC)	Current Input at AI3: 0.0149 (mA)	PT100 Resistance of Outlet: 106.9992 (Ohm)	Downstream Travel Time: 46.9494 (Micro Second)
Analog Input AI3: 0.0764	Current Input at AI4: 0.0115 (mA)	Total Travel Time: 46.9493 (Micro Second)	Output Current: 4.0002 (mA)
Analog Input AI4: 0.059	Current Input at AI5: 0.0174 (mA)	Delta Travel Time: 1.9546 (Micro Second)	Upstream Strength: 3475 (Range: 0-2047)
Downstream Strength: 3477 (Range: 0-2047)			
Ratio of travel time: 100.6944 (Range: 0-2047)			
Reynolds Number: 892.7822			

Figure 1. Display of Modbus register values.

I used Matlab to pre-process the text file provided for the task. This process transform the contents of the file into only register values. The file imported into the application and stored in an array. There are three functions created to convert raw register values into readable format. These functions include decoding REAL 4 format, decoding LONG format, and decoding INTEGER format. The decoding order for most register values correspond to higher register first and lower register second but some register values (e.g., velocity), the decoding order was lower register first and higher register second. The unit of positive accumulator was selected based on M31 table from the manual. The signal quality value extracted after dividing the decoded register value by 10 as instructed in the manual (low byte for signal quality). A conditional statement was made to present the language used in the interface. The upstream strength and downstream strength seem

to be higher than the given range of the values in the manual given in GUI. This value can be limited by adding a conditional statement. The range shown with ratio of travel time GUI is a typing mistake. The manual presents the range as (100 +- 3%). The second GUI is displayed in Figure 2.

Writing MODBUS registers:

System Password:  
\*\*\*\*\*

Hardware Password:  
\*\*\*\*\*

Calendar (date and time):  
(Format: SMHDMY)  
010213260221

Key to input:  
\*\*\*\*\*

Go to Window #:  
2

Times for the Beeper: (Max = 255)  
222

Pulses left for OCT: (Max = 65535)  
21345

Day + Hour for Auto-Save:  
(e.g: 0512H means 5th day and 12th hour)  
0512H

LCD Back-lit Lights for number of seconds: (Seconds)  
4

Figure 2. Writing Modbus register values.

There were two formats needed to be written in the Modbus registers. The first was binary coded decimal (BCD) (Registers: 49 - 56), and the second was integer (Registers: 59 – 63). There are two functions created to perform BCD conversion and extracting two bytes list. The GUI does not store the input values in variable due to the limitation of the library used. However, it can be resolved by expanding the application with other libraries. The error code bit (Register: 72) is not displayed in either of the GUI due to the lack of explanation.