Water Meter Reader Guide
CS 349: Software Engineering

Matthew Rinker, Emma Steinman, and Colin Smith
May 8, 2020

1 Purpose

The purpose of this document is to provide a detailed description of how to wire the device (to be used in conjunction with the provided pictorial wiring guide) as well as giving descriptions of how the device functions. As a high level explanation of how this water meter works, the Arduino interprets the reading given by an analog Hall Effect sensor to determine the amount of water the Water meter is recording as having been used by the building. This works because there is a magnetic wheel within the meter that spins as it records water usage.

2 Wiring

For the wiring section of this document we will cover explicitly how to wire the WiFi enabled version of the device. However, note that the only difference between the two versions is the inclusion of the ESP-01 Module.

Required Items:

- 1 Arduino Uno
- 1 16x2 LCD Dot Matrix Screen
- 1 ESP-01 WiFi Chip
- Breadboard
- 3 1kΩ Resistors
- 1 10kΩ Resistor
- 1 10kΩ Potentiometer (Optional)
- 1 Hall Effect Analog Sensor
- Wires

We will break the wiring of this device into three sections, one for each of the main devices.

1. LCD Screen

   There are 16 pins on the 16x2 LCD Screen. See the reference image in fig.1:

   The wiring of these pins is as follows:

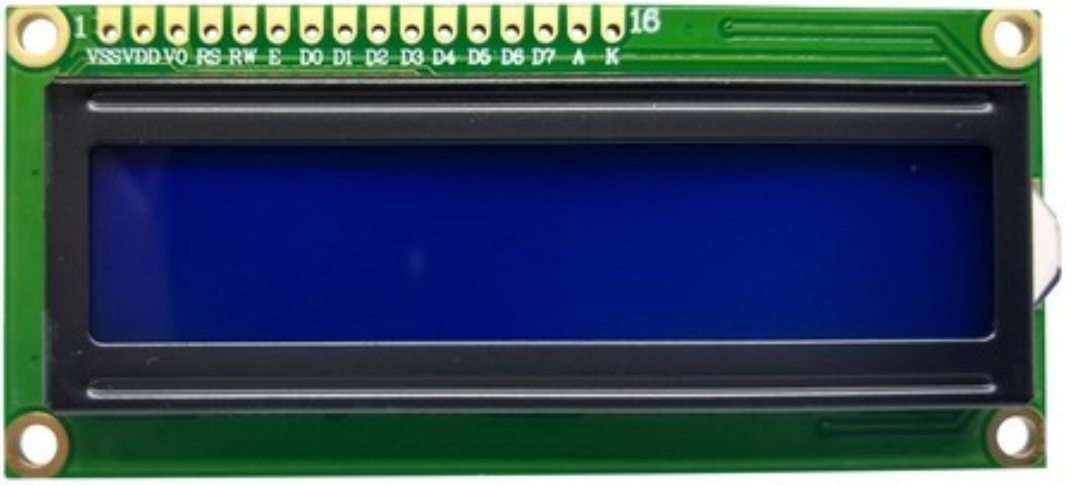

Figure 1 16x2 Dot Matrix LCD Screen

LCD	Arduino
1 (VSS)	GND
2 (VCC)	5v
3 (V0)	GND
4 (RS)	D12
5 (RW)	GND
6 (E)	D11
7 (D0)	N/A
8 (D1)	N/A
9 (D2)	N/A
10 (D3)	N/A
11 (D4)	D5
12 (D5)	D4
13 (D6)	D3
14 (D7)	D2
15 (A)	10KΩ Potentiometer
16 (K)	GND

Pin 15 is the trickiest to wire out of this set. It is connected to the 5v power through a 10kΩ Potentiometer. The Potentiometer has 3 pins. On the two outside pings 5v and GND will be connected (one on each side). Pin 15 connects to the middle pin. This allows for the voltage to be changed by twisting the knob. This is an optional step as it simply allows us to control the brightness of the screen (To bypass this you can connect Pin 15 directly to 5v).

2. Hall Effect Sensor

   The Hall Effect Sensor is quite simple to wire. There are three pins connected to the sensor. Working from left to right: The first pin is connected to 5v. The second pin is connected to GND. The Third pin is both connected

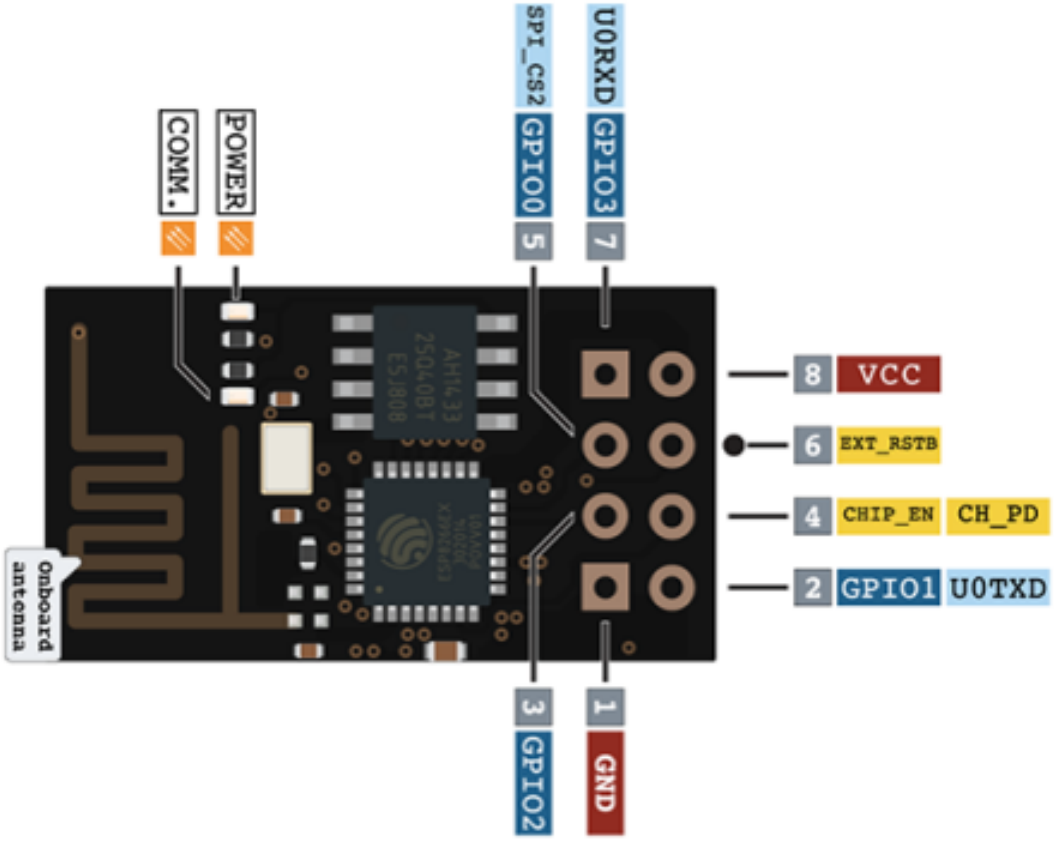

Figure 2 ESP-01 WiFi Chip

to Analog A5 as well as a 10kΩ Resistor which is connected to 5v. This is because the Analog sensor normally outputs a low value and we use the Resistor to flip that output to be better understood by the Arduino Program.

3. ESP-01

   For this explanation we will be referencing the ESP-01 Pinout Diagram in Fig.2.

ESP-01	Arduino
1 (GND)	GND
2 (GPIO1)	TX
3 (GPIO2)	N/A
4 (CH_PD)	3.3v
5 (GPIO0)	GND
6 (EXT_RTSB)	N/A
7 (GPIO3)	RX
8 (VCC)	3.3v

Most of the pins here are straightforward to wire. Supposedly we can use the 3.3v pin on the Arduino to power the 3.3v pins. However, there are sources that say that this is not possible and we must instead take a higher power source e.g. 5v or a battery and use a voltage regulator take the resistance down to 3.3v. However, since we did not have access to this equipment at the time this project ended we could not test this. Additionally, we must also transform the TX line as the TX pin on the Arduino outputs at 5v instead of the required 3.3v for the ESP-01. Any voltage higher than 3.6v can cause irreparable damage to the board. To do this the easiest way is to use 3 1kΩ resistors. Specifically, we chain two resistors on the ground line before merging that line with the TX line. We then have one more 1kΩ resistor leading to the GPIO1 pin on the ESP-01 chip. See the wiring diagram for details.

3 Code

In this section we will briefly explain the functionality of the code and how the water meter functions. To start, our code uses the WiFiEsp library for the Arduino IDE. The code to connect to a WiFi network and to print out the connection debug information were taken directly from the examples section of the Library's repository. The other section in the setup function is to define the LCD screen to which we will also be printing readouts.

Next, for the loop function we first must do the important job of factoring out the noise of the sensor. As we are using an analog sensor we will receive a lot of noise in our input. This is necessary as we know that we need the sensitivity of an analog sensor to be able to pick up the movements of the magnetic wheel within the water meter. To do this, we use two running averages to create a flat value. We essentially take the value of the next 300 data points and the last 30 of that 300 and calculate the long and short running average, The Shorter running average is subtracted from the running average. This provides a flat value which we can then use to determine if the Arduino is detecting a pulse or interfering. The last portion of the code handles the submission of the data. This is done through the third party api service Pushing Box. To do this we set up a google form that automatically dumps all entries to a google sheet. We then have a script running on the google sheets to automatically import each new line into the database. On the client side, we calculate the total gallons used since the last submission (timed to be 30 minutes apart) and then submit that value along with a constant ID we assign the Arduino. In our current test build the Arduino's deviceID is "TestArduino". This code is extremely versatile and provided we use the same database structure to manage the as of yet not obtained shark meter Data and to import that data into the database as well. .