

Developed and Created by Charles Boyer, Andrew Emerson,

Kyle Krueger, and Autumn-Storm McFaul

*An Instructive Manual on Module Creation*

***This manual is to act as a guide for the ambitious hydroponics hobbyist in mind whom wishes to create custom modules in their system. The manual includes descriptions such as where to alter the open source module code, how the code works as a whole, and the wiring of the UM232 to the desired microcontroller. All sensors and control elements will act differently and send information in various ways, so it is important to keep in mind that some alteration of the data into the microcontroller may be necessary, and that the raw data from or to the module element may have to be formatted appropriately to work with the code provided and thus the UM232.***

1

***Choosing Your Microcontroller***

The UM232 works specifically with the protocol of UART, a standard communication that is used via microcontroller. Make sure the microcontroller you choose supports UART communication and has a TX and RX pin available to send such information to the UM232.

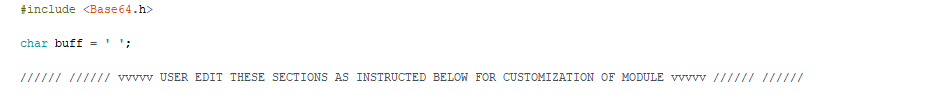
When choosing a microcontroller, the USB hub provided can draw only a maximum of 5VDC and 100mA per port. If you are unsure whether the microcontroller or module used can be powered with the USB hub on its own, it is wise to power your module using a separate power source or simply a wall outlet. As always, it is wise to check power draw and ratings before purchasing and connecting the microcontroller or device chosen.

Lastly, the code created has been written and compiled in Arduino language (C/C+ combination). Because of this, Adafruit and Arduino microcontrollers are ideal for ease of use and documentation once the code editing and compiling process begins. Keep in mind that when using non-conventional microcontrollers or microcontrollers suited for other languages, there may be less documentation on troubleshooting compile issues and more time needed to alter and customize the original code in order to appropriately send packet information to the UM232.

2

***Altering Code for Your Module***

The code provided on the source page includes character arrays which can be altered for customization of the module. The section is clearly labeled after the Base 64 library include and character buffer, indicated by “/////” before and after the section. These variables should all be carefully considered and altered, but consistent in types.



Below the above title in the code, each edited section created is explained in comment form. For titles which the user will see in the application, the name will most likely be in quotations. Keep in mind that some characters will not show on the application, and that there is a limit to how many characters can be displayed across the screen. Some of the fields will be explained in more depth below:

**ModuleName:** Simply the name of the module title displayed

**Alias:** create this as a unique number to name this module in the system this keeps the system from sending the name numerous times and wasting time and space in memory. Numbers that act as an alias within the system must all be unique. Make sure to label the module itself as this alias so future users and module creators can avoid using the module alias in the same system. Create the module so you may change the alias at any time in case another system module has the same alias that you or the user wishes to use.

**NumberofValues:** Unfortunately, the Hydroforya module code cannot currently support more than one value within the module. However, the main system can support a dynamic number of values which all are treated as separate systems.

**AssocType:** This is the association type which categorizes the “auto-connect” portion of Hydroforya. Two module pairs, such as a lamp and a photosensor, which will connect via the central system and then control and affect one another via the control system leveling, must have the same association type in order to use the auto-connect features. Other modules that are not paired must not have the same module association type. For instance, a pH solution should not be paired with a photosensor, lest the pH level change via the tolerances requested by the control system when light into the photosensor is altered. If you notice a module is changing at the same time as another module when not desired, this may be the issue. Check the code to see if association numbers are the same, and if they are, make them differ.

**ValueName1:** The name of the unitsof the value being monitored.

**ValueSize:** This will help the system size the value and is what makes the size of the module system dynamic in terms of data given in a packet. Follow the directions above this field carefully. Make sure the unit value from the raw data you want to send to the control system will not exceed the numbers given in the directions. If unsure of the value range, guess a higher number rather than a lower number.

**MaxRange/MinRange:** a cruder measurement of range. Again, be sure that the system will never exceed these numbers (guess much lower for min range, guess much higher for max range), otherwise packets may not send properly through the system.

**TolRes:** This is the resolution of the value, or the tolerance of the system. This value is what the system can increment by. A “touchy” system, or more sensitive system, will have a lower number. A harsh or crude system, like an LED, will have a high number for the system. If you have a system with a wide range, and set the tolerance very low, keep in mind the system will change very slowly. For module pairings like pH or a pump, this may be necessary, but may be inconvenient for modules like a grow lamp or temperature.

**InitialValue:** This is the setpoint the control system at. This is helpful to a control element if the sensor is not paired or the system is not paired immediately. It will set the control element to a default if no association is created or the app is for some reason not working properly with the system.

**DesiredValue:** This value will eventually, but again, is a failsafe in case the association is not created with the system. This differs from the initial value in that some systems need a “startup value” or a value which is different from the desired that must be slowly incremented to the desired, like that of a pump. Of course, this may be the same number as the initial value in some cases. The desired value can always be changed via the phone application.

**MinTime:** This is the *time* in which the tolres can be changed. This indicates what the system can handle in terms of how fast the levels can change. For instance, a system will look similar when changing if the tolerance resolution is 10 and the min time is 50 milliseconds verses a system which has a tolerance resolution of 1 and the min time is 500 milliseconds.

**RecTime:** If minimum time is time that the system *can* be changed, the recommended time is the time it *should* be changed. These may be the same time.

**AppET:** This represents the application element shown to alter the desired number. If you would like only an on/off switch which does not alter the specific levels, then a button can suffice. If a more precise level as a range of numbers is required, a slider may be used. If it is desired to control the level to specific numbers in the system, then a raw field may be used so the user my manually type the number to control the control element by.

**Step:** Just as the recommended time differs from the minimum time, the tolerance differs from the steps of the system. The step the system will change by is the “recommended tolerance” or thus what increment the system should safely change as. It is best if the step size is higher than the tolerance.

You will see in the template code that there is a section that supports a hypothetical module sensor:



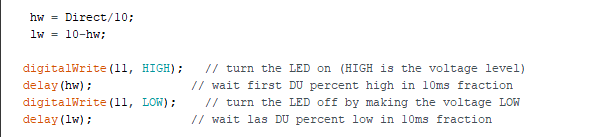
…



The InitialValue set in the system should be a number that is dynamic in the code, and thus one an incoming packet or sensor read should clearly change. This is true for changing the value in the system to allow for control. To ensure this, make sure the value of the sensor/control system that you wish to send via packet is set as the variable “Data”, as shown above with the calculated resistor value from the voltage analog reading.

Keep in mind that the code to turn the raw sensor data into a char array format to send via the packet to the UM232 will vary from system to system. This is also true for the values coming from the central system to a module which is a control element. It is the task of the module creator to create code which will take in the sensor values as a number and specify the leveling these values for the fields above.

Similarly, the control element has a variable named “Direct” which should be dynamic in the system. The code shown below shows how this code is dynamic and may be altered. In this case, a PWM wave is simply created using delays (of course this would be optimized with interrupts, but this method will still work with the system):



…

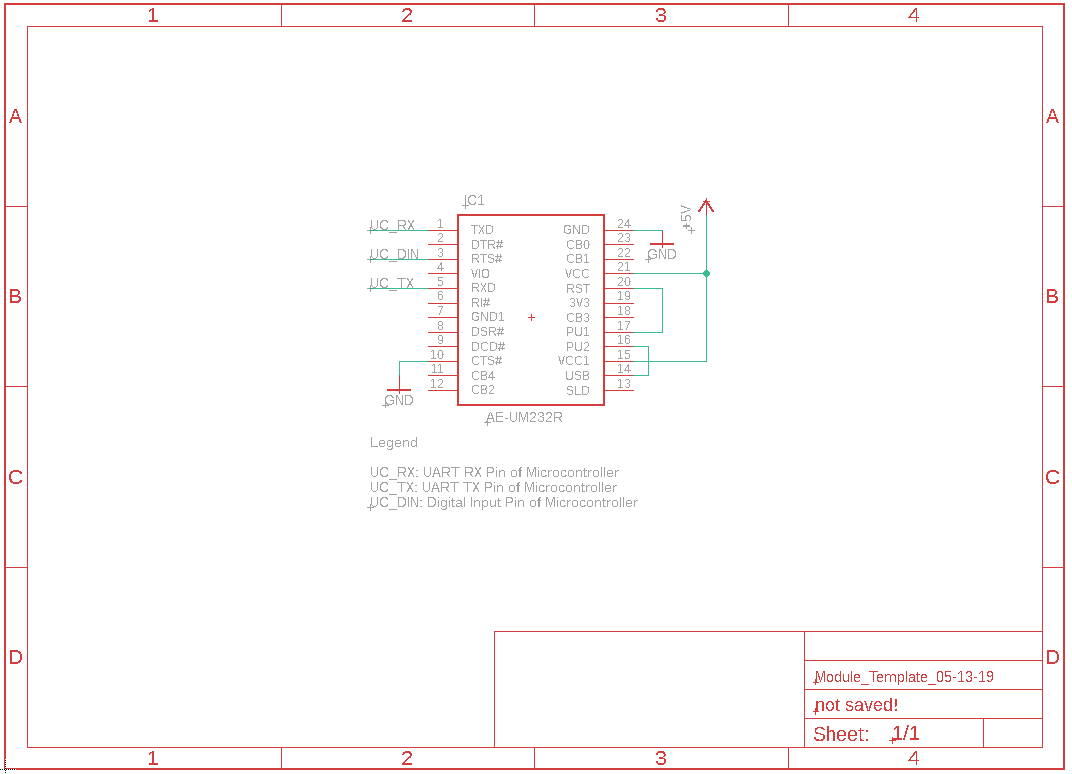


When altering the variables in the input fields (within the fields indicated by ///// placements) the values that it is indicative of are “Direct” and “Data”. Keep this in mind when changing tolerances, etc., and the raw data code.

If you find during the compiling of the code that a line other than the lines above are causing an error, check to make sure the base 64 library is properly in your folder, and that the microcontroller is properly installed onto your computer. The Hydroforya team recommends that, unless you have professional background in programming, other fields besides the lines in code mentioned above should not be altered during the code customization step. This keeps more advanced compiling errors from arising and focuses troubleshooting to more surface level, syntactical, and electromechanical errors. Provide link to gethub of base64.

3

***Connecting the Microcontroller to the UM232***

Below is the schematic for the UM232 in the Hydroforya system. The UM232 will be powered via the USB connection with the central system (the USB hub of the central system) and thus the only power to the microcontroller of your choice will be the 5VDC of the VCC of the UM232. Otherwise, if the module has a separate power supply, you may choose to power the UM232 via the module’s power supply. The UM232 documentation is provided to assist with alternate setups such as this.

*Schematic of UM232 of A Module for Hydroforya, with UART connection to Microcontroller*

The CTS# connection goes to GND to set the UM232R in UART mode. TXD pin of the UM232R may then be connected to the UART RX pin of any microcontroller with UART and vise-versa (RXD of UM232 to TX of the microcontroller). The RTS# pin may be connected to the microcontroller those who wish to add a directional signal along with the code. It is not required with the available code but is available for those whom wish to utilize this method.