Start Guide

QEC-R11HU9S : EtherCAT Gateway RS232/485 + Keypad + LCM + MPG with 86EVA and ArduBlock

86Duino Coding IDE 501

EtherCAT Library

(Version 2.0)

Revision

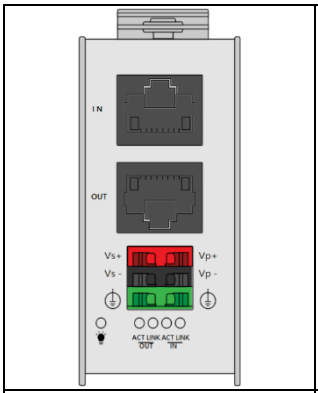
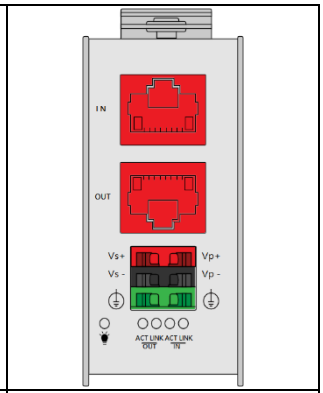
Date	Version	Description
2024/2/2	Version1.0	New release.
2024/10/8	Version1.1	Split the development steps into two documents.
2025/11/6	Version2.0	Combine all functions of HU9S in one document.

Preface

In this guide, we will show you how to use the EtherCAT MDevice **QEC-M-01** and the **QEC-R11HU9S** series (EtherCAT SubDevice, with RS232/485, Keypad, LCM, and MPG Hand wheel).

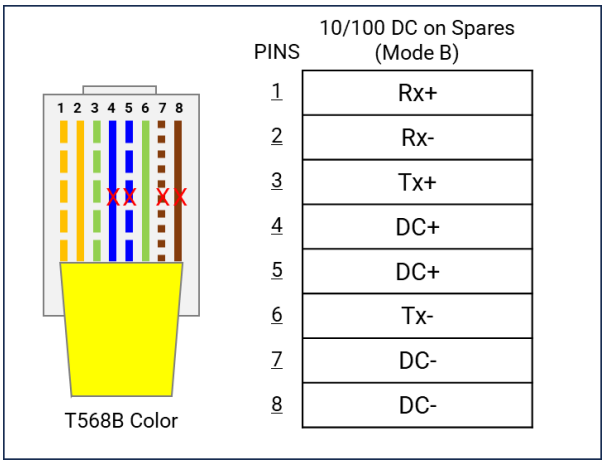
Notes QEC's PoE (Power over Ethernet)

In QEC product installations, users can easily distinguish between PoE and non-PoE: if the RJ45 house is red, it is PoE type, and if the RJ45 house is black, it is non-PoE type.

	
Non-PoE type	PoE type

PoE (Power over Ethernet) is a function that delivers power over the network. QEC can be equipped with an optional PoE function to reduce cabling. In practice, PoE is selected based on system equipment, so please pay attention to the following points while evaluating and testing:

1. The PoE function of QEC is different and incompatible with EtherCAT P, and the PoE function of QEC is based on PoE Type B, and the pin functions are as follows:



PINS	10/100 DC on Spares (Mode B)
1	Rx+
2	Rx-
3	Tx+
4	DC+
5	DC+
6	Tx-
7	DC-
8	DC-

2. When connecting PoE and non-PoE devices, make sure to disconnect Ethernet cables at pins 4, 5, 7, and 8 (e.g., when a PoE-supported QEC EtherCAT MDevice connects with a third-party EtherCAT SubDevice).

3. QEC's PoE power supply is up to 24V/3A.

1. Connection and wiring hardware

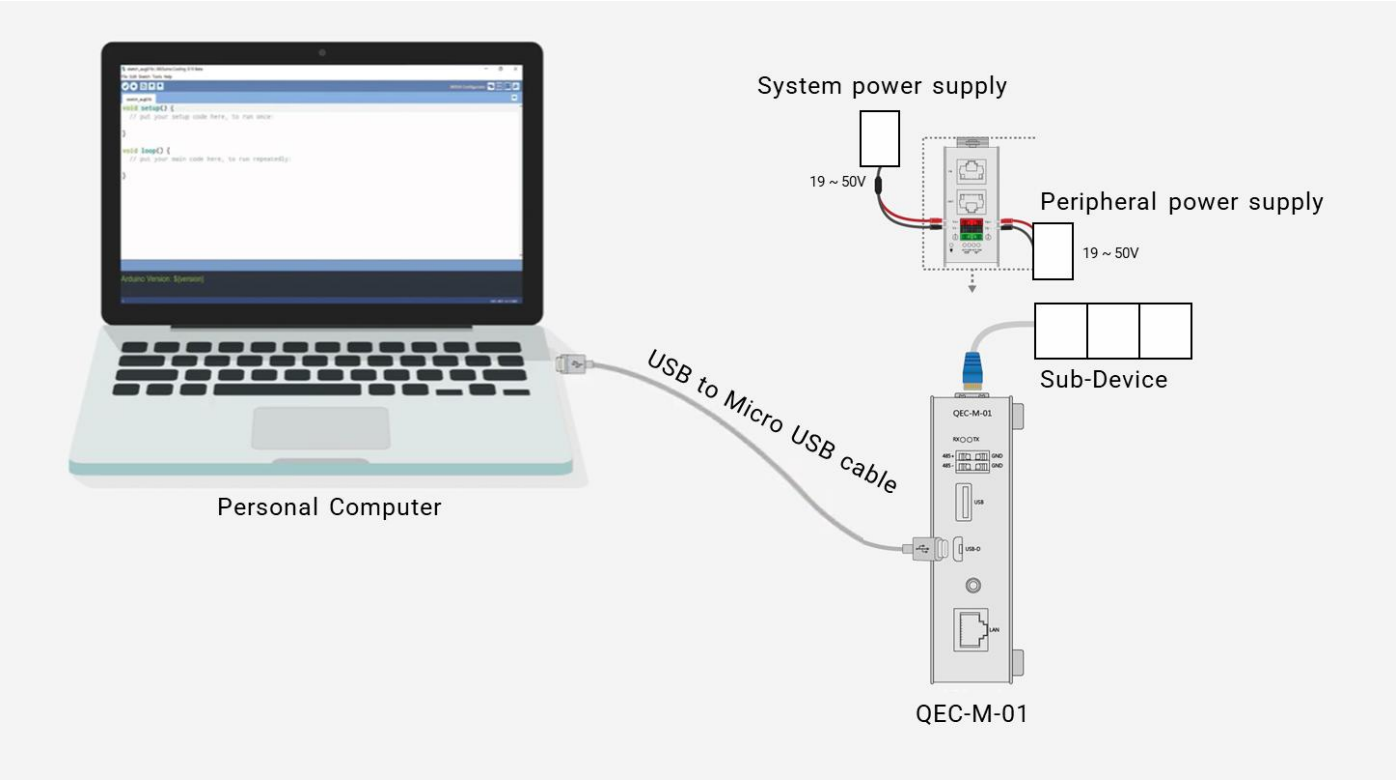
The following devices are used here:

1. QEC-M-01 (EtherCAT MDevice)
2. QEC-R11HU9S series (EtherCAT SubDevice, with RS232/485, Keypad, LCM, and MPG Hand wheel)
3. 24VDC power supply & EU-type terminal cable & LAN cable

1.1 QEC-M-01

QEC-M-01 is an EtherCAT MDevice.

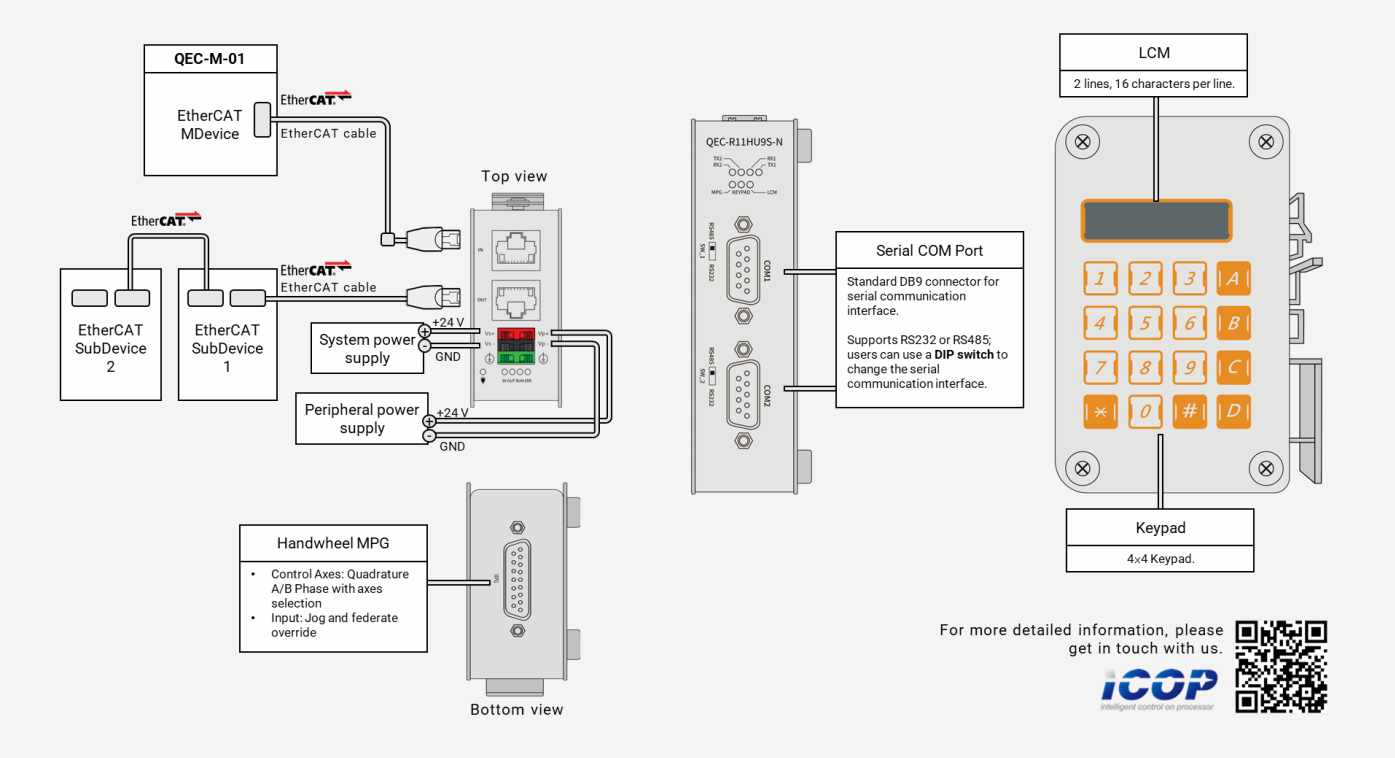
1. Power Supply: Connect to Vs+/Vs- and Vp+/Vp- power supplies via EU terminals for 24V power.
2. EtherCAT Connection: Using the EtherCAT Out port (On the top side) connected to the EtherCAT In port of EtherCAT SubDevice via RJ45 cable.



1.2 QEC-R11HU9S

The **QEC-R11HU9S** is an EtherCAT SubDevice module with RS232/485, Keypad, LCM, and MPG Hand wheel.

The diagram shows a typical wiring example with a QEC MDevice and an EtherCAT network.

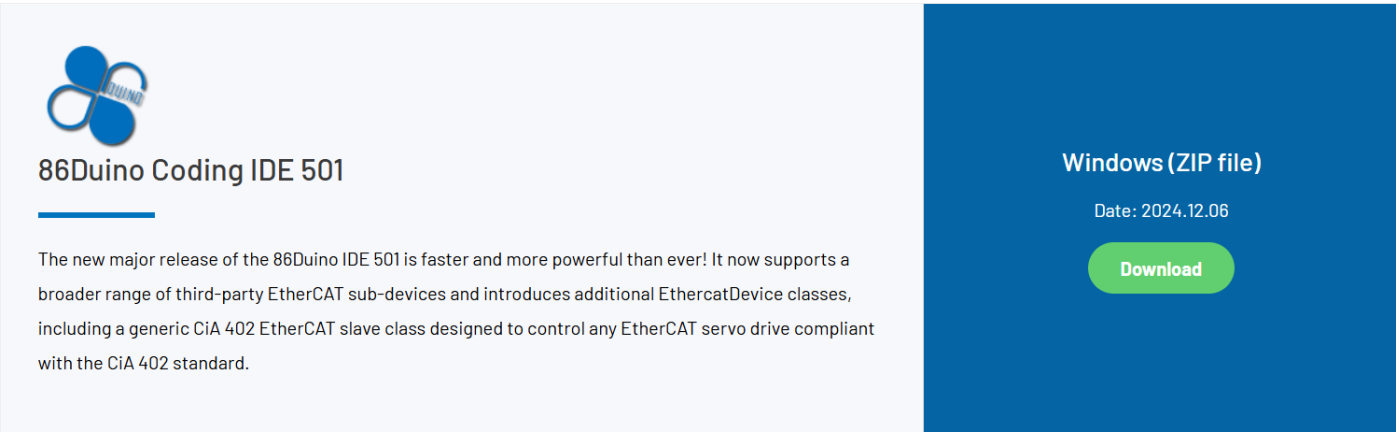


Connections are grouped by function:

- EtherCAT: MDevice → IN; OUT for daisy-chain
- Power & Grounding:
  - VS+/VS−: system power +24 V/GND
  - VP+/VP−: field I/O power +24 V/GND
- Serial COM: DB9; DIP switch selects RS-232/RS-485
  For RS-485, use termination/bias and a shared reference
  - Data transfer rate (bps): 2400,4800,9600,14400,19200,38400,57600,115200
  - Data width (bit): 5/6/7/8
  - Hardware Flow Control: CTS/RTS
- Handwheel MPG
  - Control Axes: Quadrature A/B Phase with axes selection
  - Input: Jog and federate override
- Keypad: Matrix 4×4
- LCM: 2 lines, 16 characters per line
- Indicators: PWR, RUN, LINK, ERROR, TX/RX/MPG/KEYPAD/LCM status

2. Software/Development Environment

Download 86duino IDE from https://www.qec.tw/software/.



After downloading, please unzip the downloaded zip file, no additional software installation is required, just double-click 86duino.exe to start the IDE.

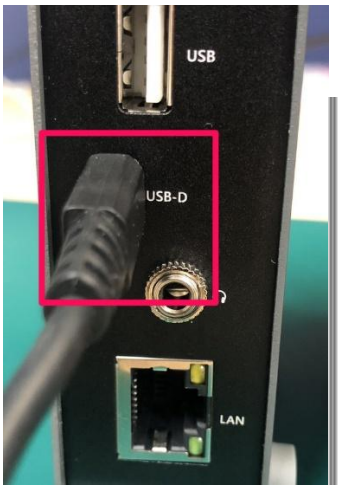
*__Note__: If Windows displays a warning, click Details once and then click the Continue Run button once.

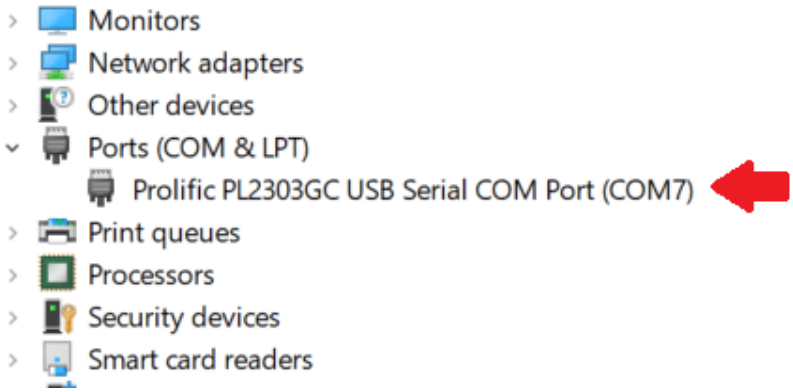
86Duino Coding IDE 501+ looks like below.

3. Connect to PC and set up the environment

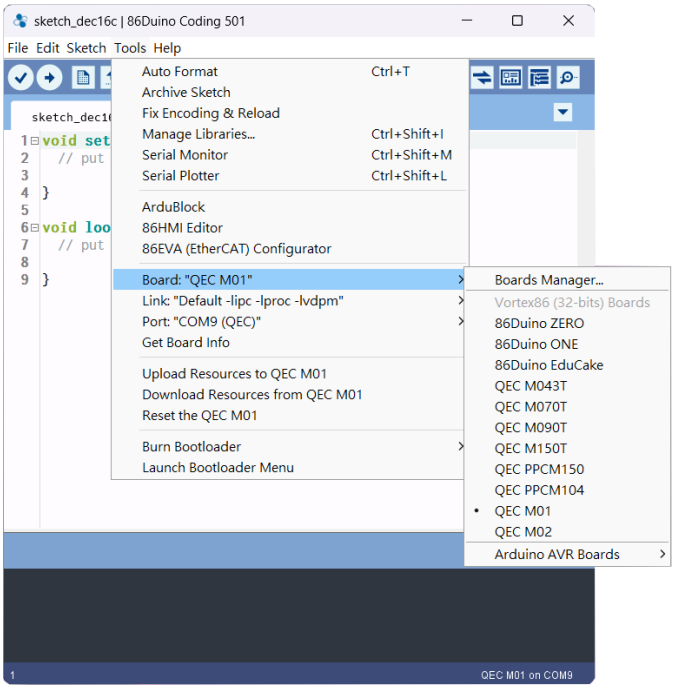
Follow the steps below to set up the environment:

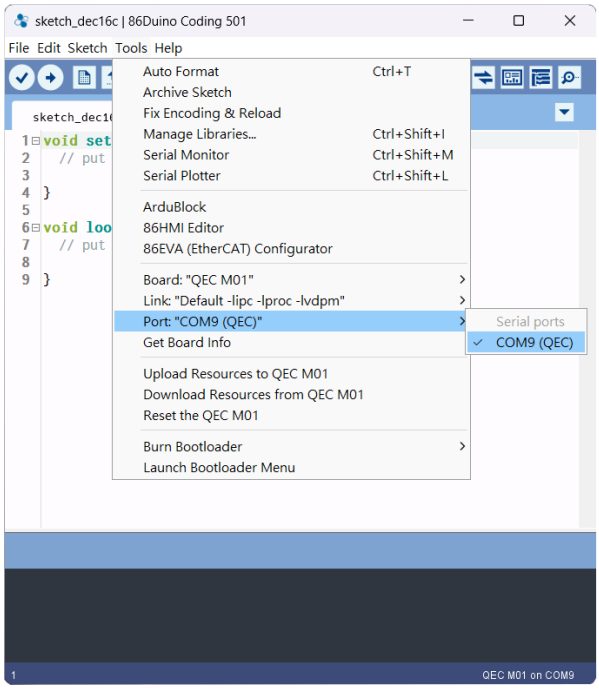
1. Connect the QEC-M-01 to your PC via a Micro USB to USB cable (86Duino IDE installed).
2. Turn on the QEC power.
3. Open “**Device Manager**” (select in the menu after pressing Win+X) ->” **Ports (COM & LPT)**” in your PC and expand the ports; you should see that the “**Prolific PL2303GC USB Serial COM Port (COMx)**” is detected; if not, you will need to install the required drivers.
   (For Windows PL2303 driver, you can download here)





4. Open the 86Duino IDE.
5. Select the correct board: In the IDE's menu, select “**Tools**” > “**Board**” > “**QEC-M01**” (or the QEC MDevice model you use).
6. Select Port: In the IDE's menu, select “**Tools**” > “**Port**” and select the USB port to connect to the QEC MDevice (in this case, COM9 (QEC)).





4. Use 86EVA with ArduBlock

This example shows how to operate the EtherCAT MDevice (QEC-M-01) and the QEC-R11HU9S series (EtherCAT SubDevice, with RS232/485, Keypad, LCM, and MPG Hand wheel) through the 86Duino IDE's graphical low-code programming tool, 86EVA.

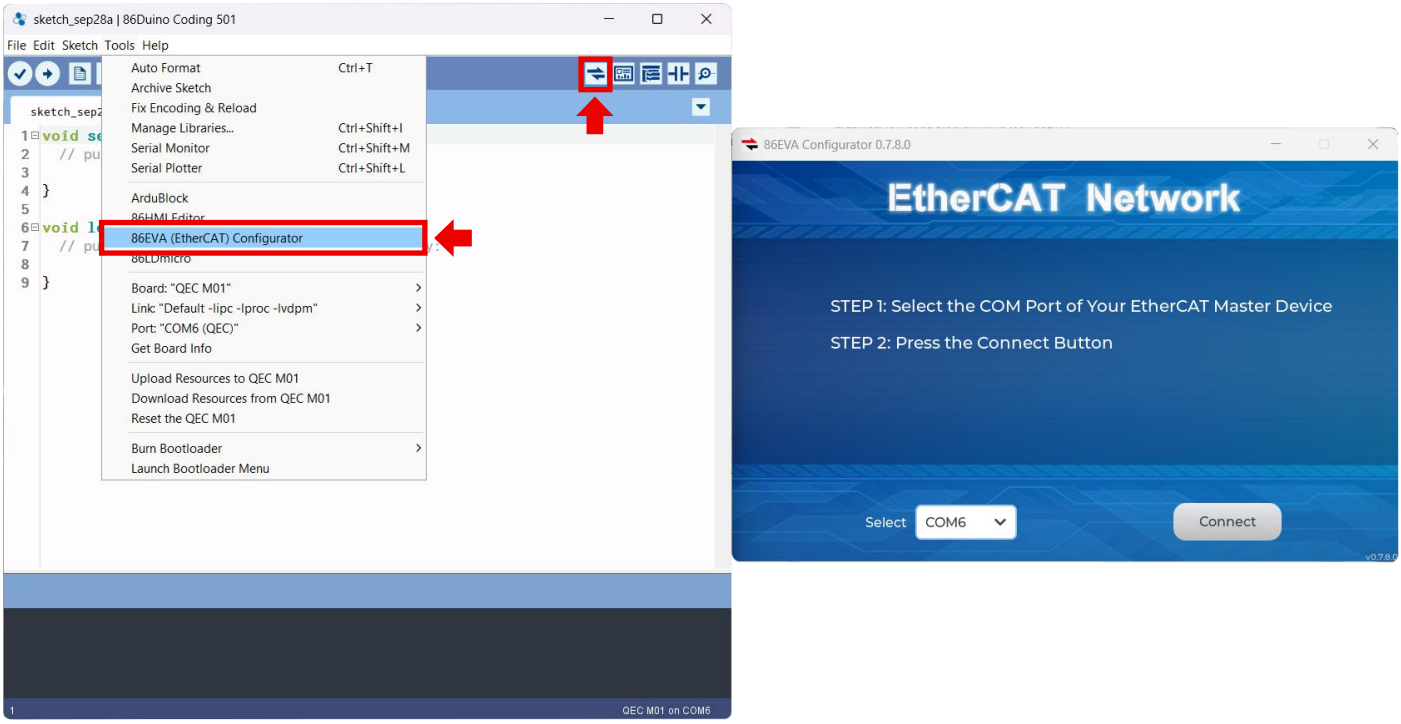
Software Tools Description:

- **86EVA (EVA, EtherCAT-Based Virtual Arduino)**:
  is a graphical EtherCAT configuration tool based on the EtherCAT Library in the 86Duino IDE and is one of the development kits for 86Duino.
- **ArduBlock:**
  is a graphical interface for programming and IO control. It is third-party software that belongs to Arduino IDE, developed by David Li, a Shanghai-based creator, and must be attached to the IDE to operate. ArduBlock is a software that converts graphical blocks into code and eventually generates the main program to 86Duino Coding IDE, then compiles and uploads it.

We will present the Serial COM Port (RS232/RS485), Keypad and LCM, and MPG hand wheel usage in the following section by steps.

Step 1: Turn on 86EVA and scan

The 86EVA tool can be opened via the following buttons.



Please select the correct COM port and then click the “**Connect**” button.

Once you have confirmed that the correct COM port has been selected of QEC-M-01, press the Connect button to start scanning the EtherCAT network.

The connected devices will be displayed after the EtherCAT network has been scanned.

Step 2: Set the parameters

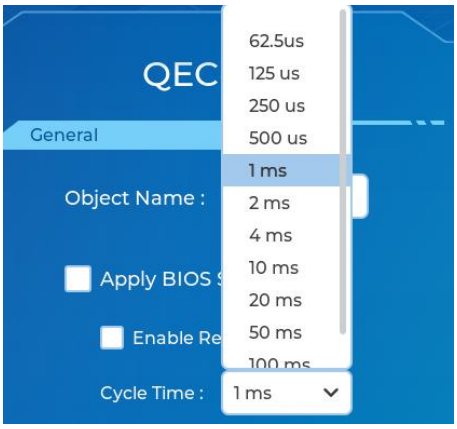
Press twice on the scanned device image to enter the corresponding parameter setting screen.

Step 2.1: QEC-M-01

Press twice on the image of the QEC-M-01 to see the parameter settings.

Please check the following configures.

1. Turn off the "**Apply BIOS Settings**".
2. Select "**1ms**" to the Cycle Time.



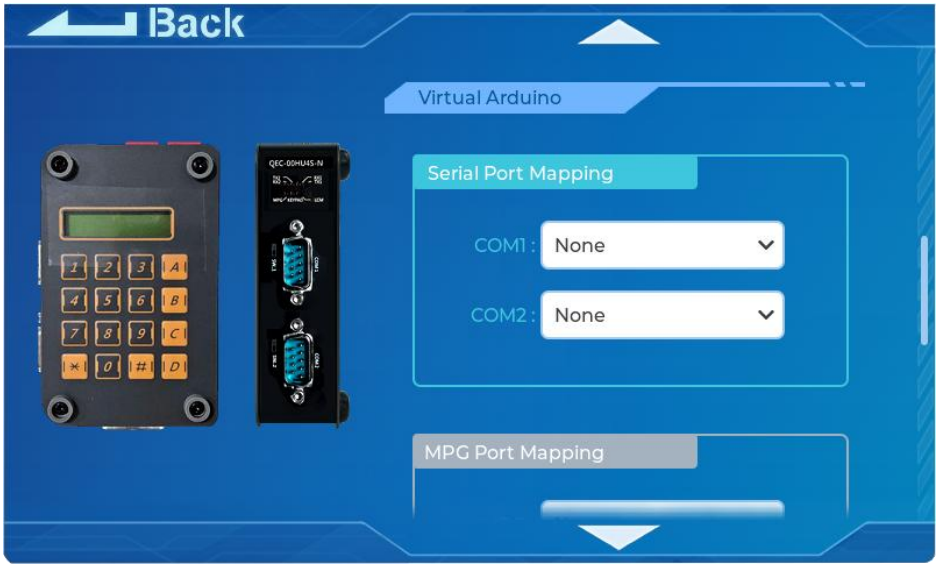
Click "**Back**" in the upper left corner to return.

Step 2.2: QEC-R11HU9S

Press twice on the image of the QEC-R11HU9S to see the parameter settings.

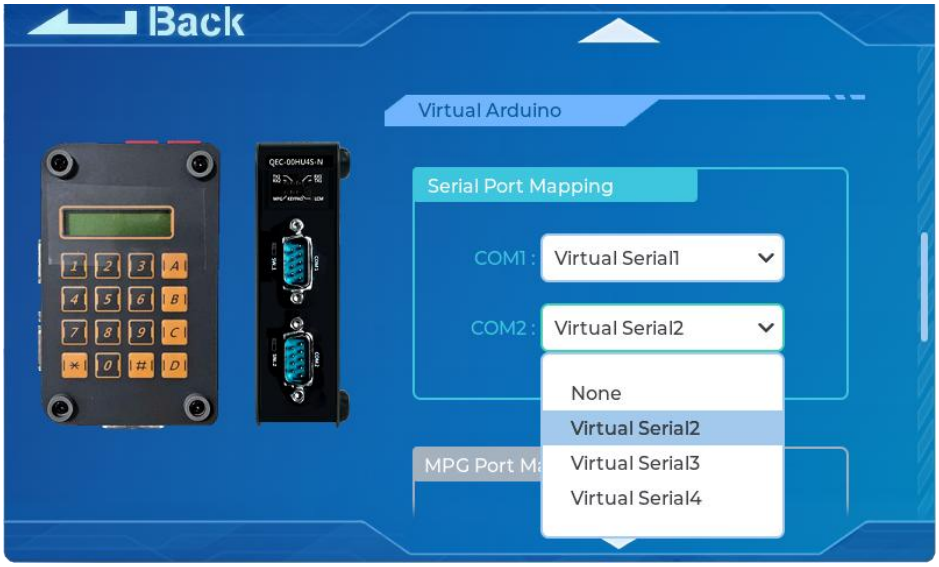
The page will show the Object Name, Alias Address, Vendor ID, Product Code, Virtual Arduino Mapping, and Virtual Servo Configuration parameters.

Continue down to the "**Virtual Arduino**" area.

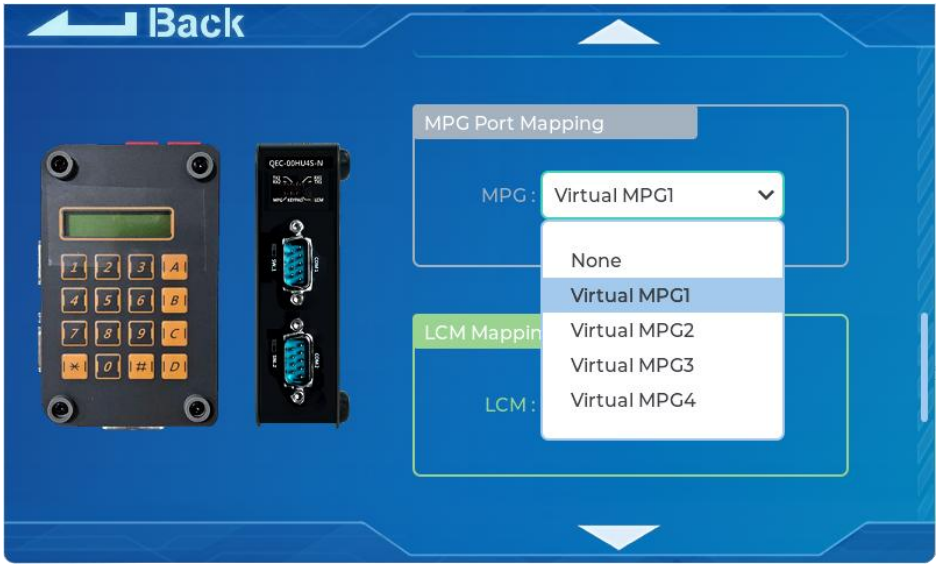


There are five areas of the Virtual Arduino in the QEC-R11HU9S: **Serial Port Mapping**, **MPG Port Mapping**, **LCM Mapping**, **Keypad Mapping**, and **Buzzer Pin Mapping**.

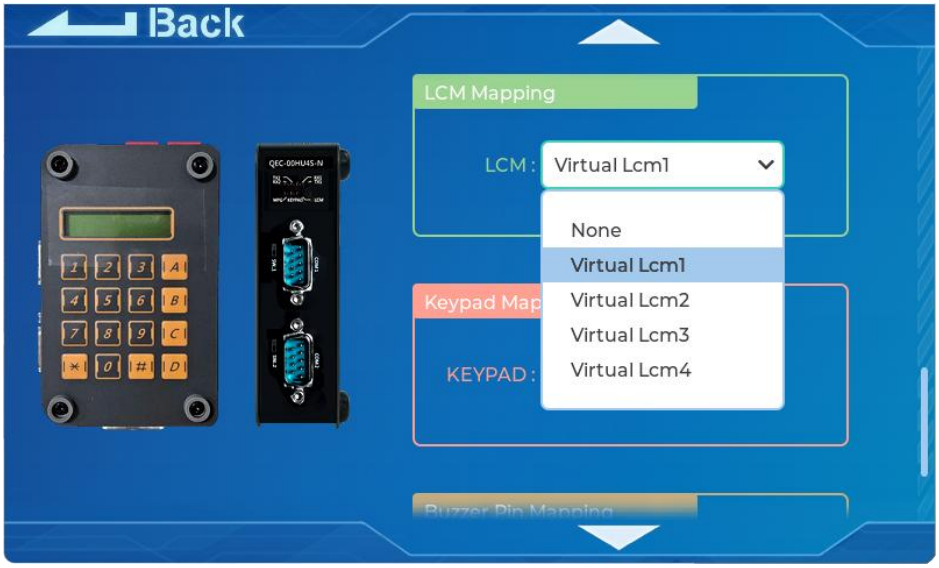
First, we select "**Virtual Serial1/2**" in the drop-down box of "**COM1**" and "**COM2**" in the "**Serial Port Mapping**" area.



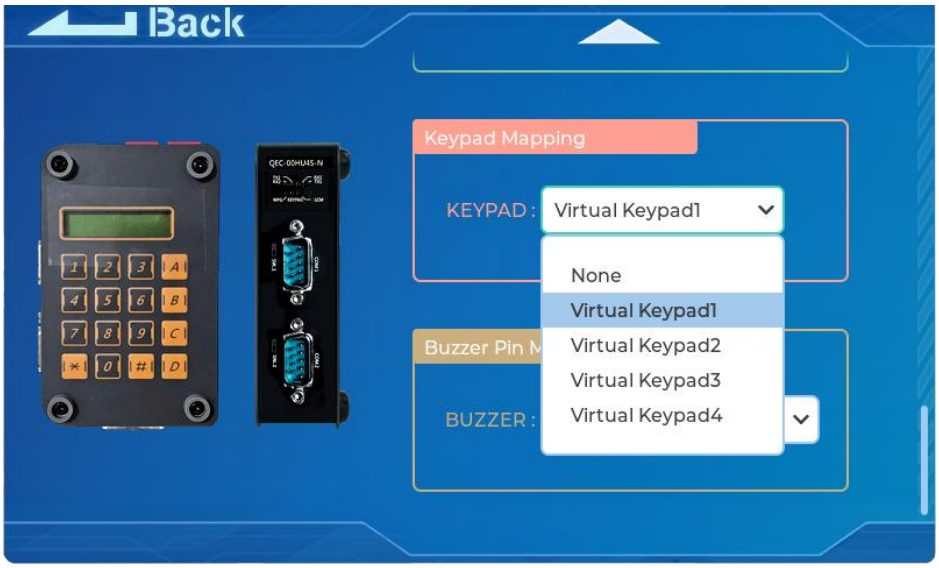
Next, we select "**Virtual MPG1**" in the drop-down box of "**MPG**" in the "**MPG Port Mapping**" area.



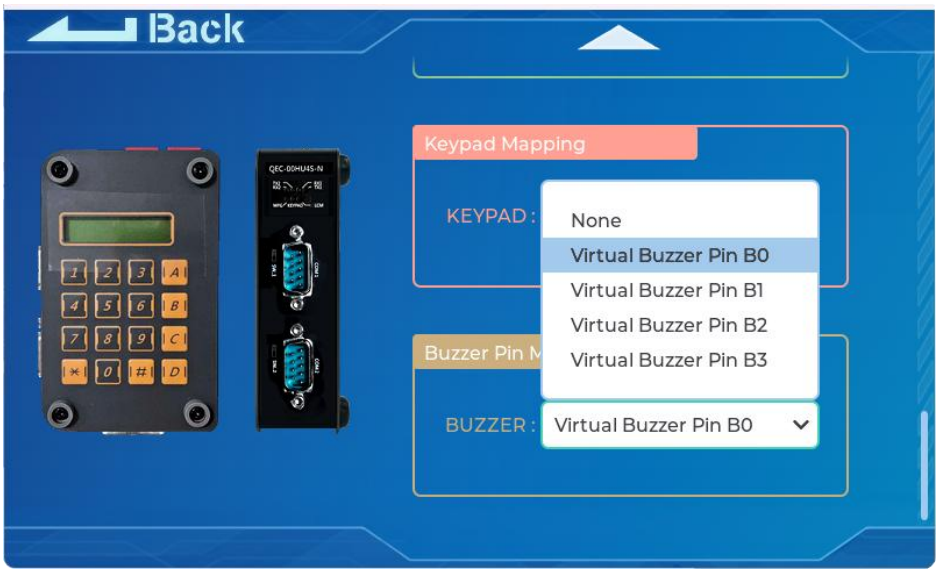
We select "**Virtual Lcm1**" in the drop-down box of "**Lcm**" in the "**LCM Mapping**" area.



We select "**Virtual Keypad1**" in the drop-down box of "**KEYPAD**" in the "**Keypad Mapping**" area.



We select "**Virtual Buzzer Pin B0**" in the drop-down box of "**BUZZER**" in the "**Buzzer Pin Mapping**" area.

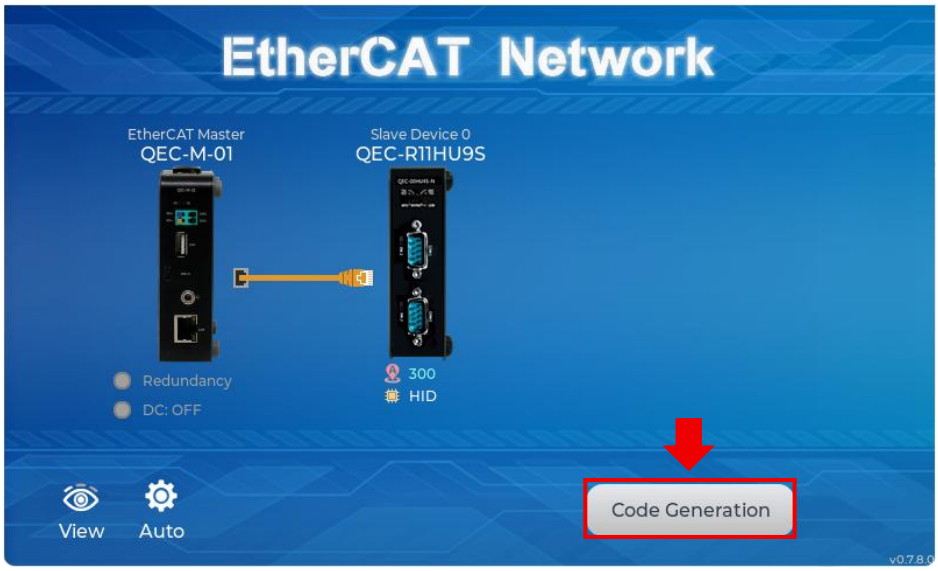


All these settings and mappings are for ArduBlock tool configuration.

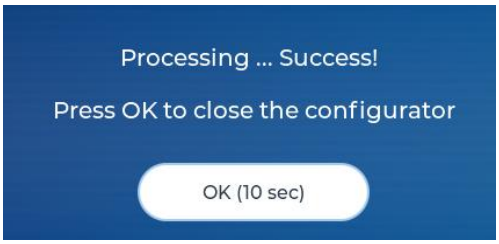
Next, click "**Back**" in the upper left corner to return.

Step 3: Generate the code

Once you've set your device's parameters, go back to the home screen and press the "**Code Generation**" button in the bottom right corner.

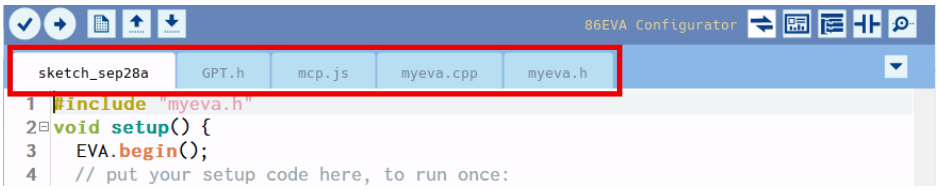


When you're done, double-click the "**OK**" button to turn off 86EVA, or it will close in 10 seconds.



The generated code and files are as follows:

- sketch_sep10b: Main Project (.ino, depending on your project name)
- myeva.cpp: C++ program code of 86EVA
- myeva.h: Header file of 86EVA



***Additional note:**

After 86EVA generates code, the following code will be automatically generated in the main program (.ino), and any of them missing will cause 86EVA not to work.

1. `#include “myeva.h”` : Include EVA Header file
2. `EVA.begin();` in `setup()` : Initialize the EVA function

Step 4: Turn on ArduBlock and setup

Before operating the EtherCAT network, you must configure it once. The process should be from Pre-OP to OP mode in EtherCAT devices. 86EVA will automatically handle the EtherCAT State Machine in the background.

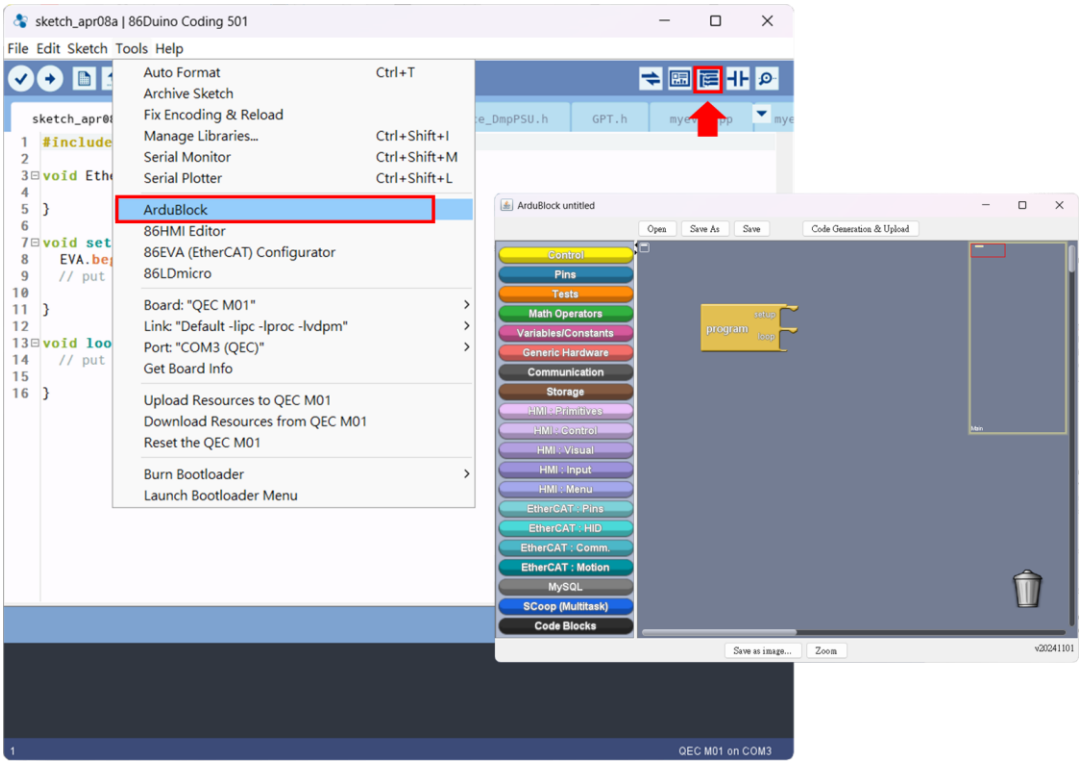
The programming code from 86EVA are set as the following by default:

- QEC-R11HU9S module: `EthercatDevice_QECR11HU9S` object.
- EtherCAT mode: ECAT_SYNC.

And here is the setting by users:

- EtherCAT Cycle time: 1 millisecond.
- The `EthercatMaster` object (“master”) represents the QEC-M-01, while the `EthercatDevice_QECR11HU9S` object (“slave0”) represents the QEC-R11HU9S module.

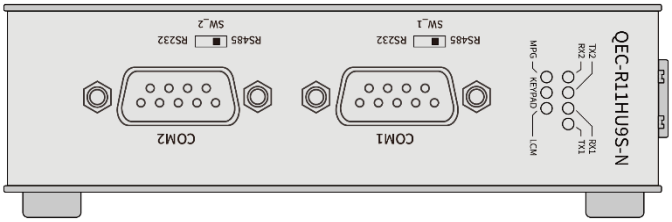
Next, after the 86EVA sets the Virtual Arduino Pins, we can open ArduBlock.



We will present the Serial COM Port (RS232/RS485), Keypad and LCM, and MPG hand wheel usage in the following section by steps.

Step 4.1: Serial COM Port (RS232/485)

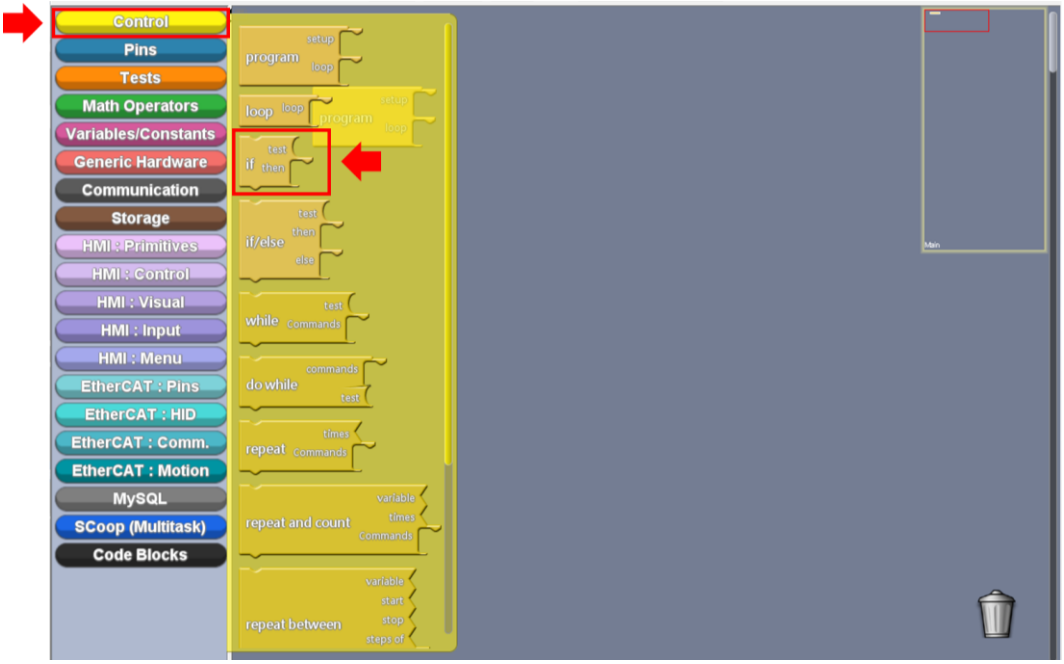
In this section, we will read the data from the Serial Monitor in the 86Duino IDE and transfer it from COM1 (RS-232) to COM2 (RS-232). After COM2 receives the data, we print it on the Serial Monitor.



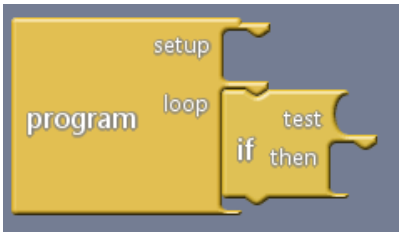
Wiring

- Use the front DB9 connector (see diagram for pinout).
- Connect COM1 to COM2; you can connect to the external device's TX ↔ RX, RX ↔ TX, and GND ↔ GND.
- Keep cable length reasonable; share a common reference (GND) with the device.

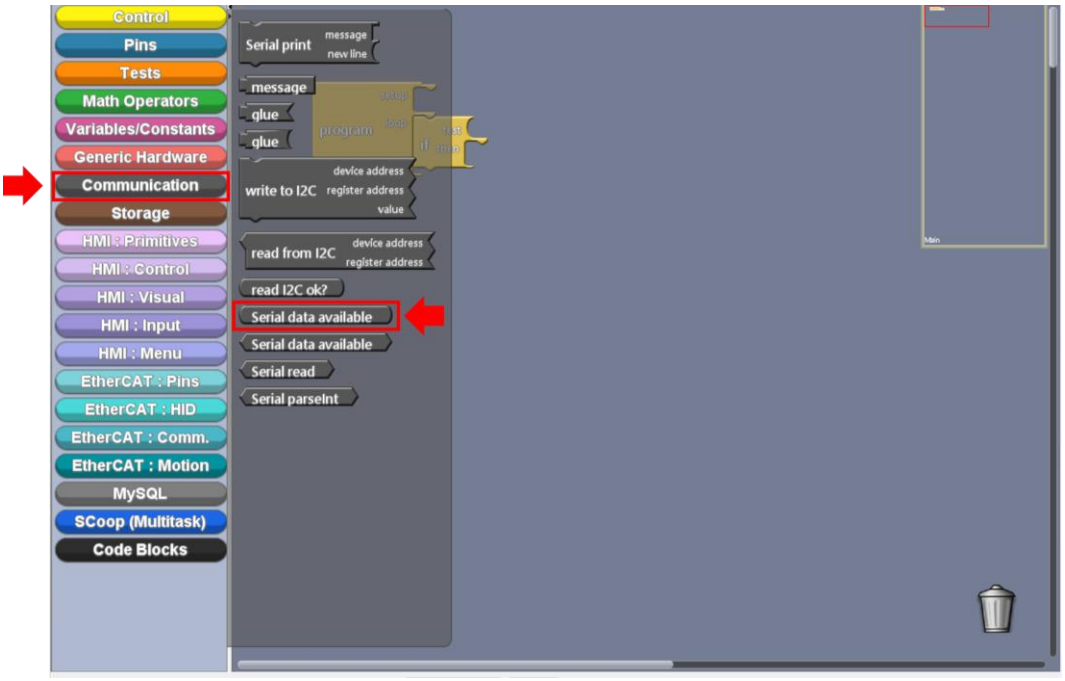
First, we use the "**if** " block from the "**Control**" class to the program's main loop to determine whether the Serial Monitor is available.



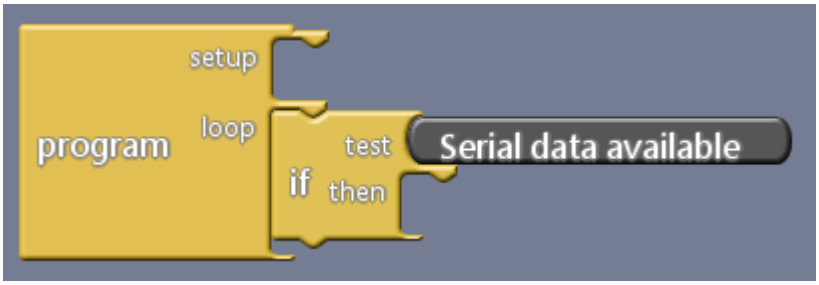
Put under the "**loop**" area in the "**program**" block.



Then, we use the "**Serial data available**" block from the "**Communication**" class.



And put in the "**if** " block "**test**" area.



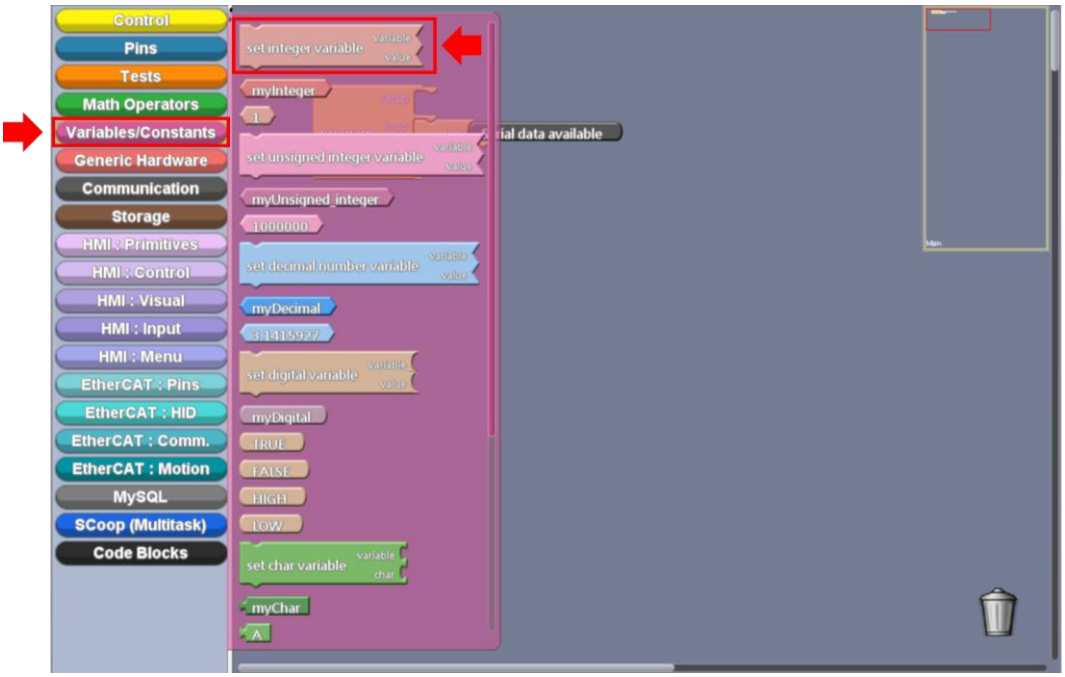
The "**if** " will determine the "**Serial data available**". If it returns true, it will execute the program.

***Note:**

The default baud-rate of Serial monitor is 9600.

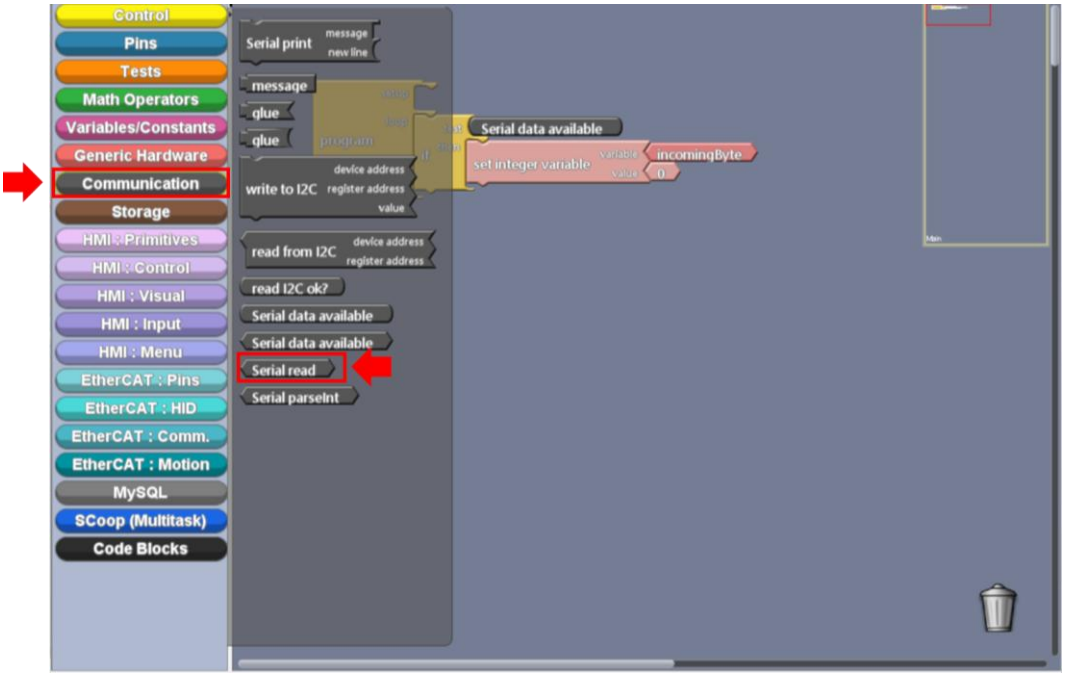
In the then area of “**if**” block, we will start processing the HID RS232 communication.

We use the “**set integer variable**” block from the “**Variables/Constants**” class to create an input value received from Serial Monitor.



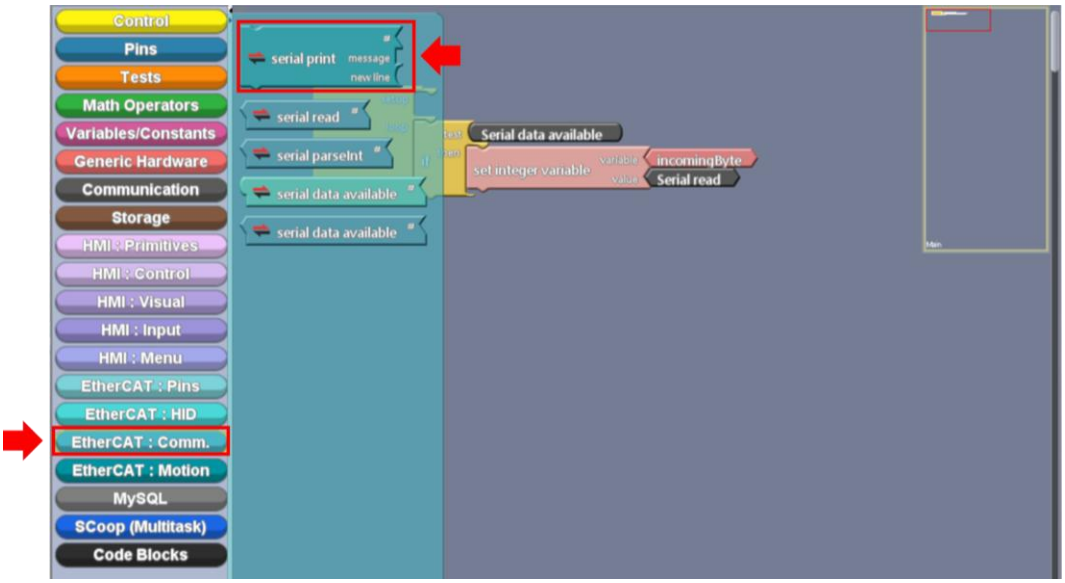
Put in the "**if** " block “**then**” area and change the variable name to “**incomingByte**”.

We use the “**Serial read**” block from the “**Communication**” class, and put it in the value area of “**set integer variable**” block.

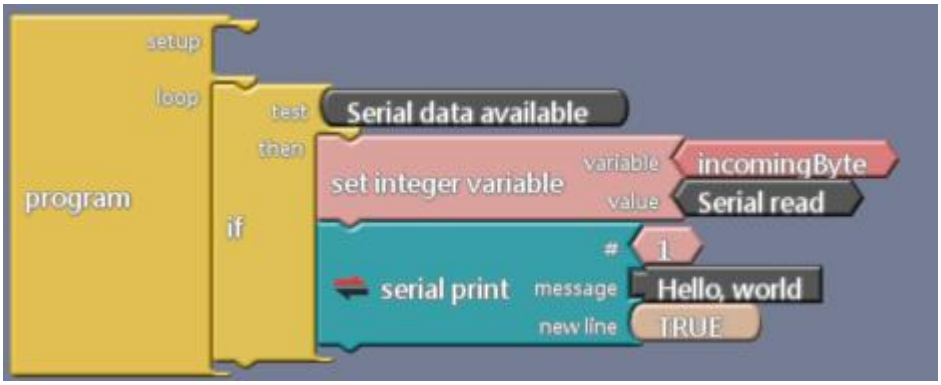


Like this.

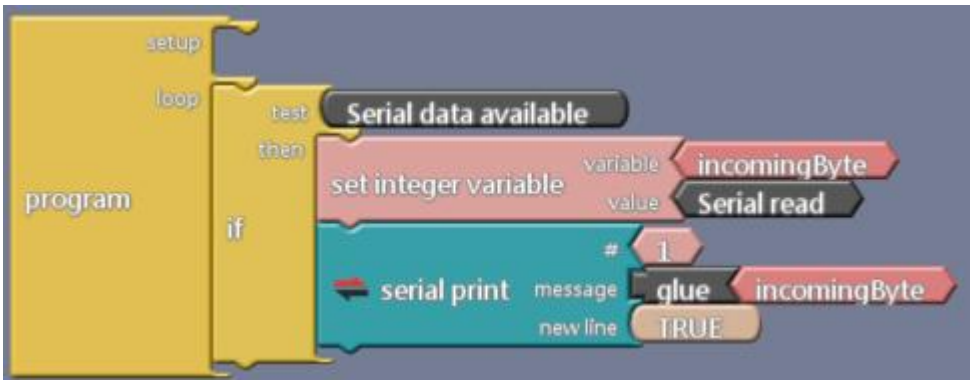
Next, we use the "**serial print**" block from "**EtherCAT : Comm.**" Class, to send data from Serial Monitor input through "**Virtual Serial Port1**" (we select "**Virtual Serial Port1**" for HU9S COM1).



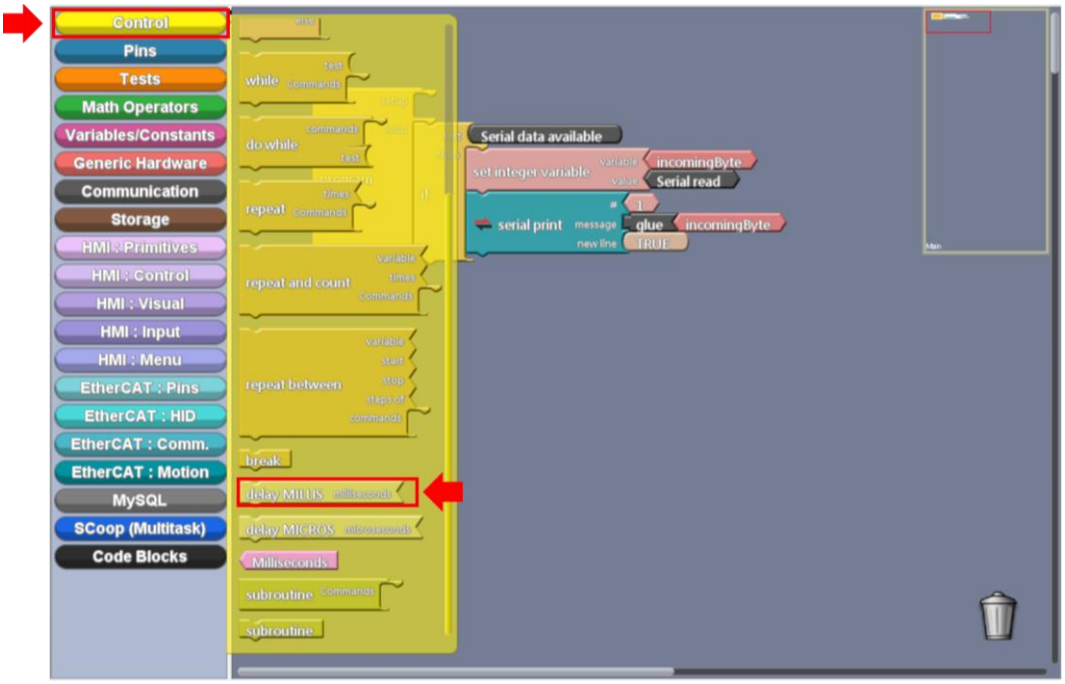
Like this.



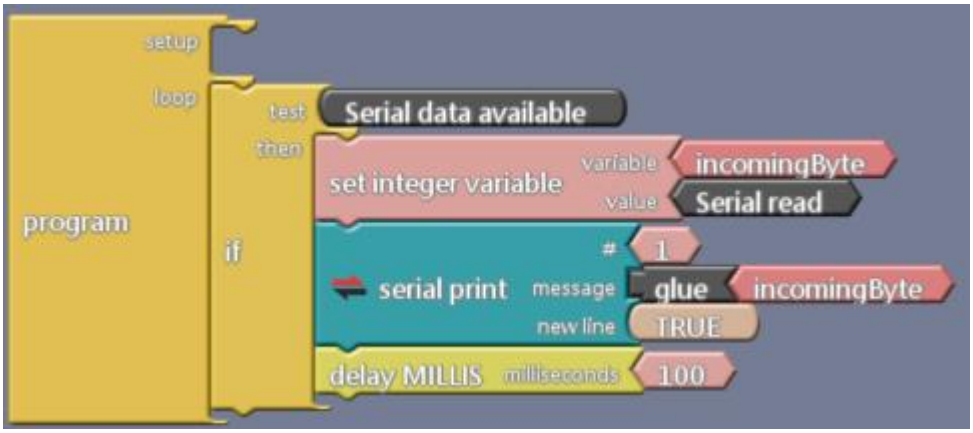
And we need to change the message content from "**Hello, world**" to the variable "**incomingByte**". We use "**glue**" block from "**Communication**" Class to glue the "**incomingByte**" to the message area of "**serial print**".



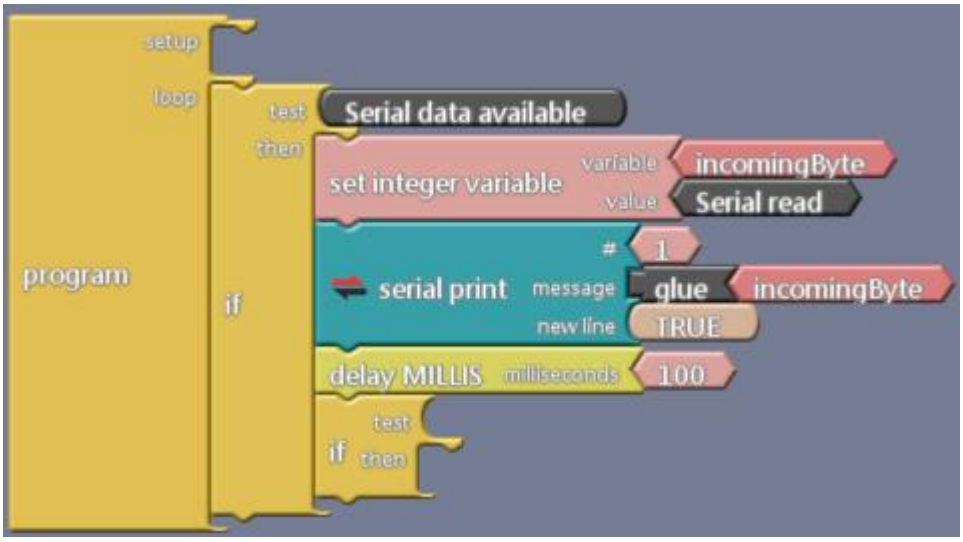
Because we need to wait for the serial communication time, we put a "**delay MILLIS**" block in the "**Control**" class.



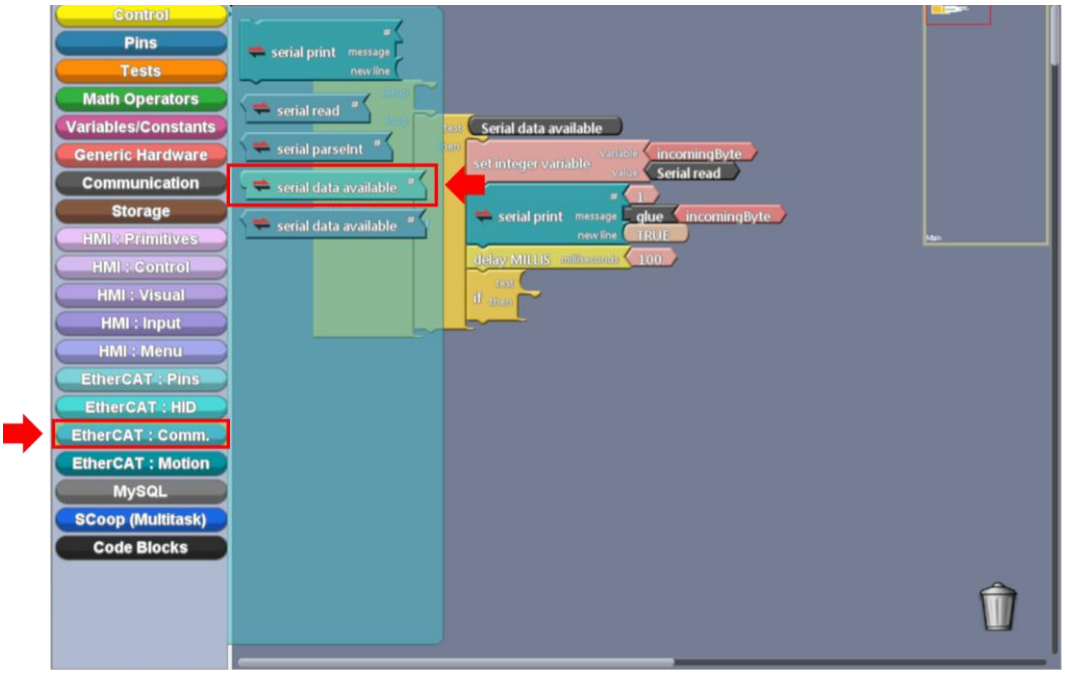
In this case, we delay for 100 milliseconds.



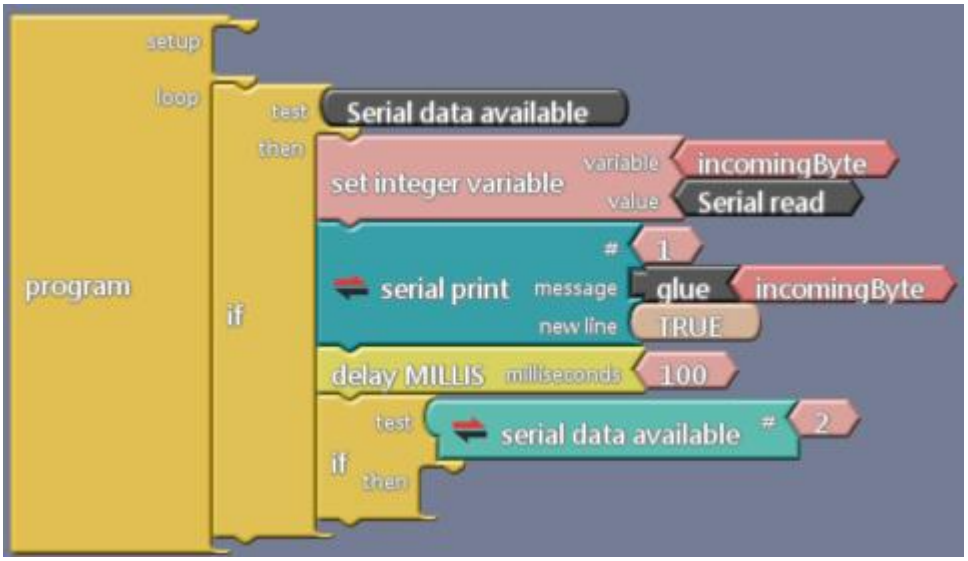
Then, we must determine whether the "**Virtual Serial Port2**" (we select **"Virtual Serial Port2"** for HU9S COM2) received the data from the "**Virtual Serial Port1**". So, we create an "**if**" block after the "**delay MILLIS**" block.



In the test area in the **"if"** block, we use **"serial data available"** block from **"EtherCAT : Comm."** class.

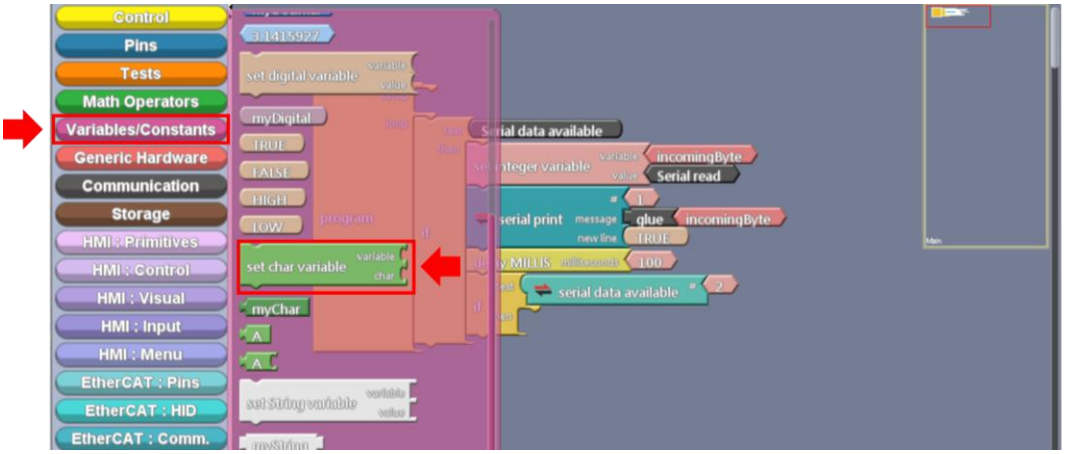


And set the "**#**" number to 2, to let "**serial data available**" to read the "**Virtual Serial Port 2**".



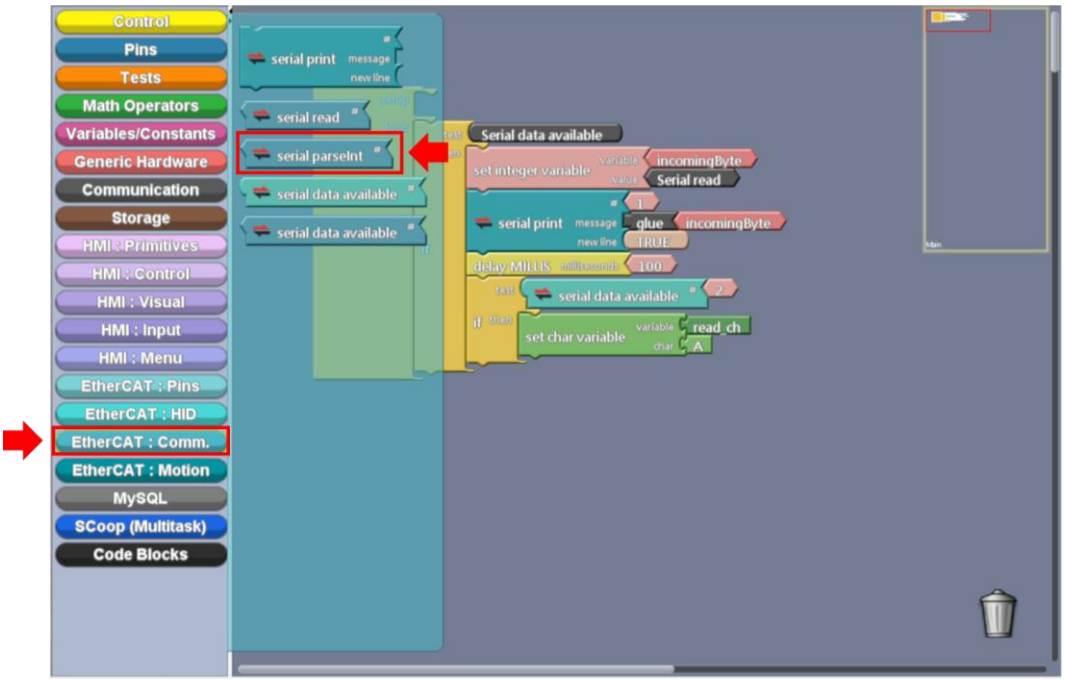
The "**if**" will determine whether the "**Virtual Serial Port2**" is available. If it returns true, it will execute the program.

In the "**then**" area of "**if**" block, we will read the "**Virtual Serial Port2**" received data, and set it in a char variable, which will call "**read_ch**".
First, we use the "**set char variable**" block from "**Variables/Constants**" Class, and set the variable name to "**read_ch**".

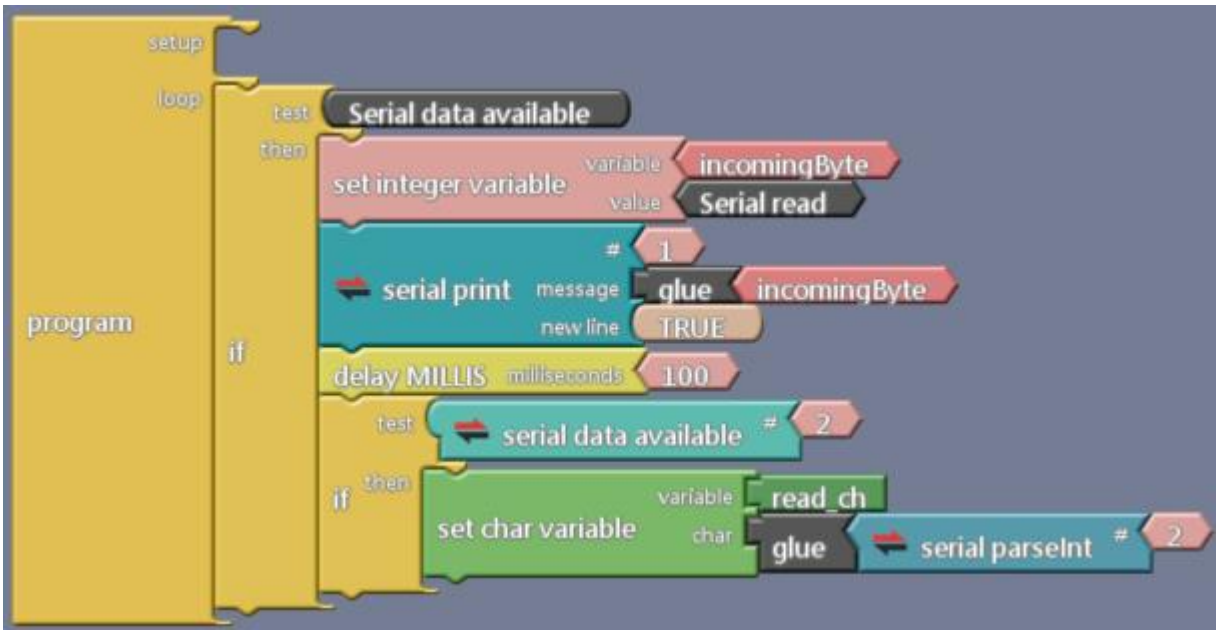


Like this.

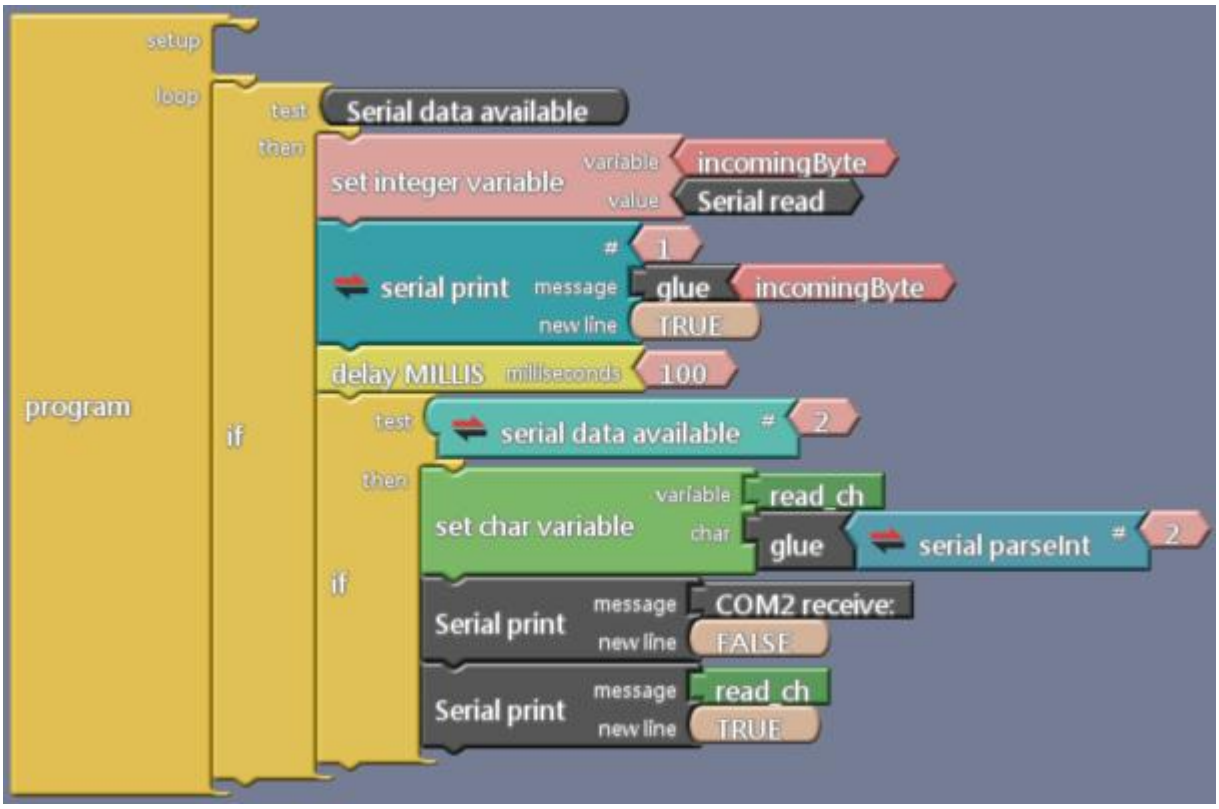
And we use the “**glue**” block from “**Communication**” class to glue the received data from “**Virtual Serial Port2**” by using the “**serial parseInt**" block from "**EtherCAT : Comm**" class to the char area of “**set char variable**”.



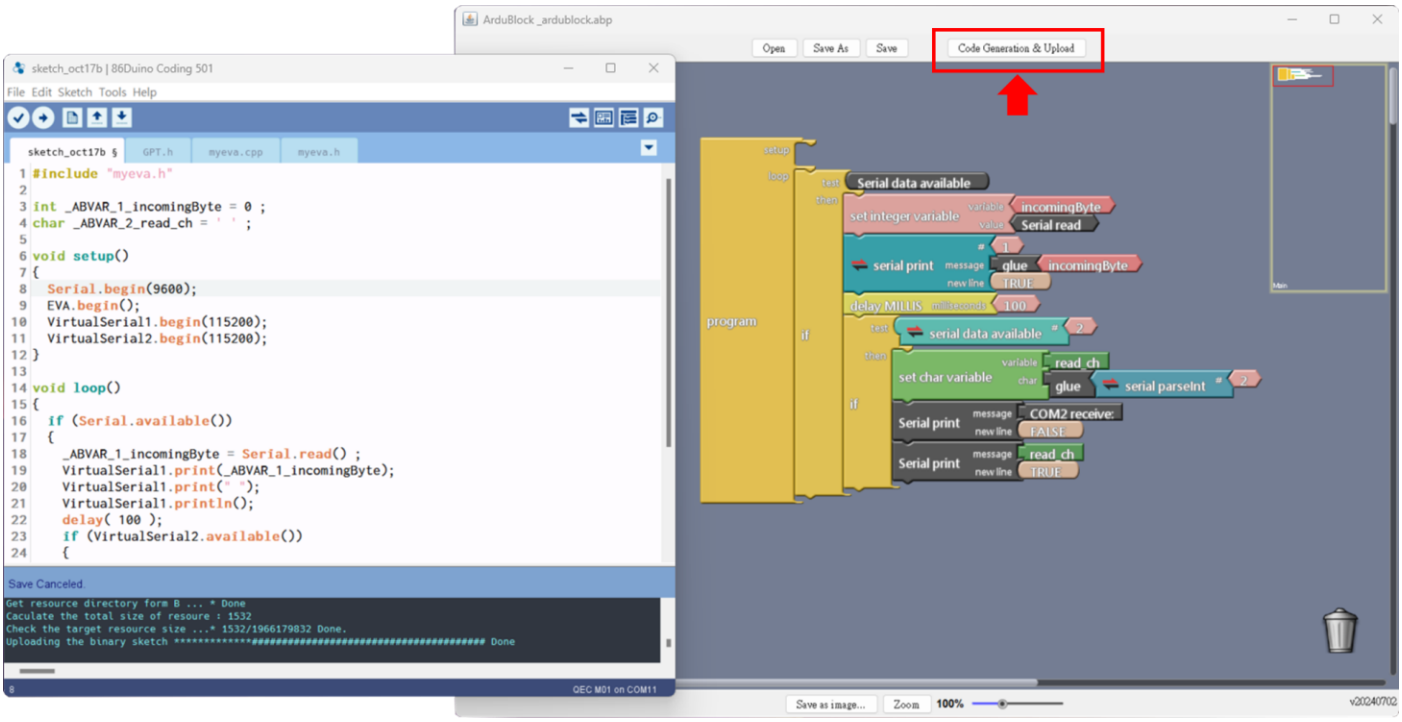
Like this.



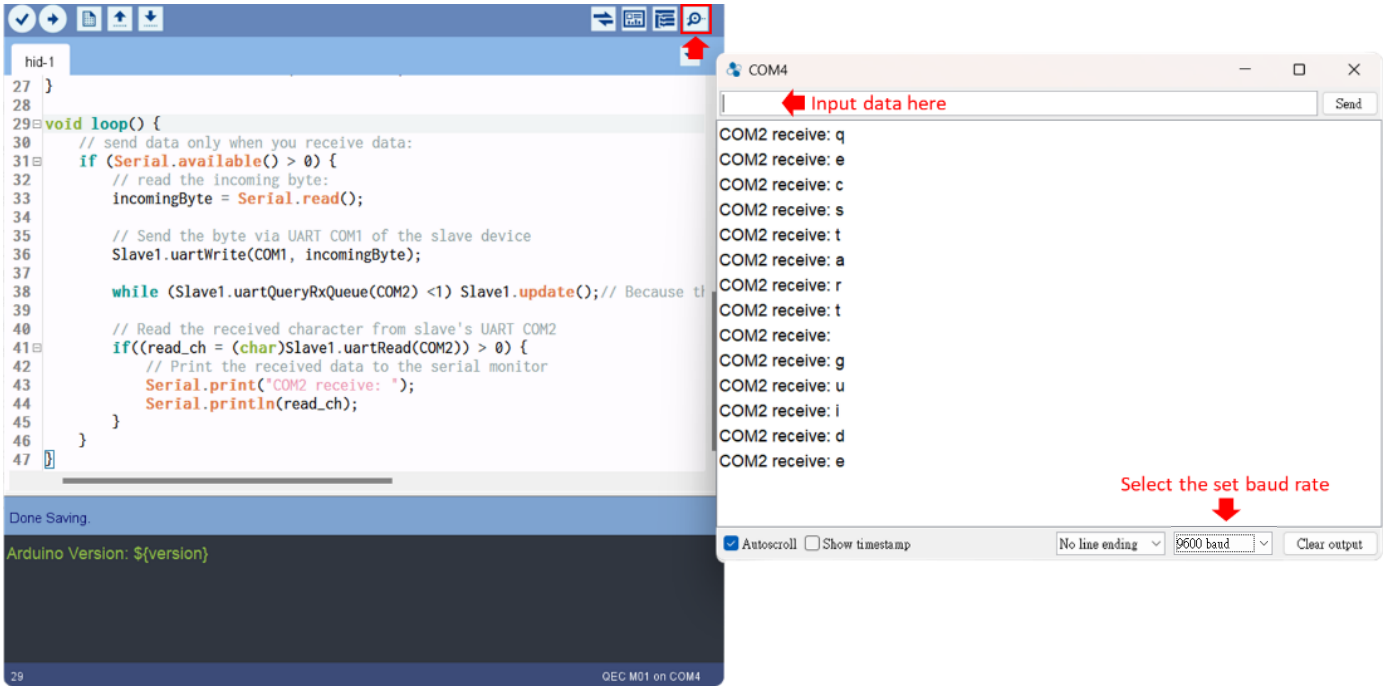
We use the “**Serial print**” block to print out the char variable “**read_ch**” (the data received from “**Virtual Serial Port2**”) in Serial Monitor.



After you finish, click the “**Code Generation & Upload**” button, and ArduBlock will automatically generate the program code and upload it to the QEC-M-01.

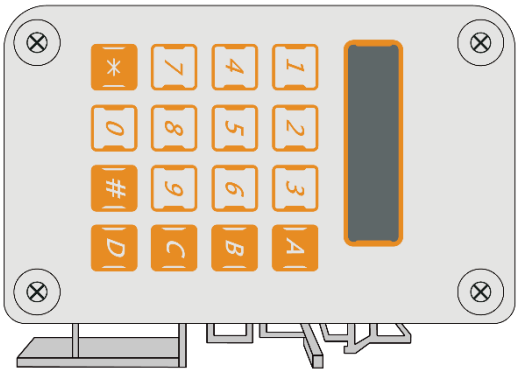


After uploading, you can input a number or letter to the Serial Monitor in 86Duino IDE. All data will transfer from COM1 to COM2. After COM2 receives the data, we print it on the Serial Monitor, as in the image below.



Step 4.2: Keypad + LCM + Buzzer

In this section, we read keypad input and print it at specific positions on the LCM. The buzzer beeps whenever a key is pressed.

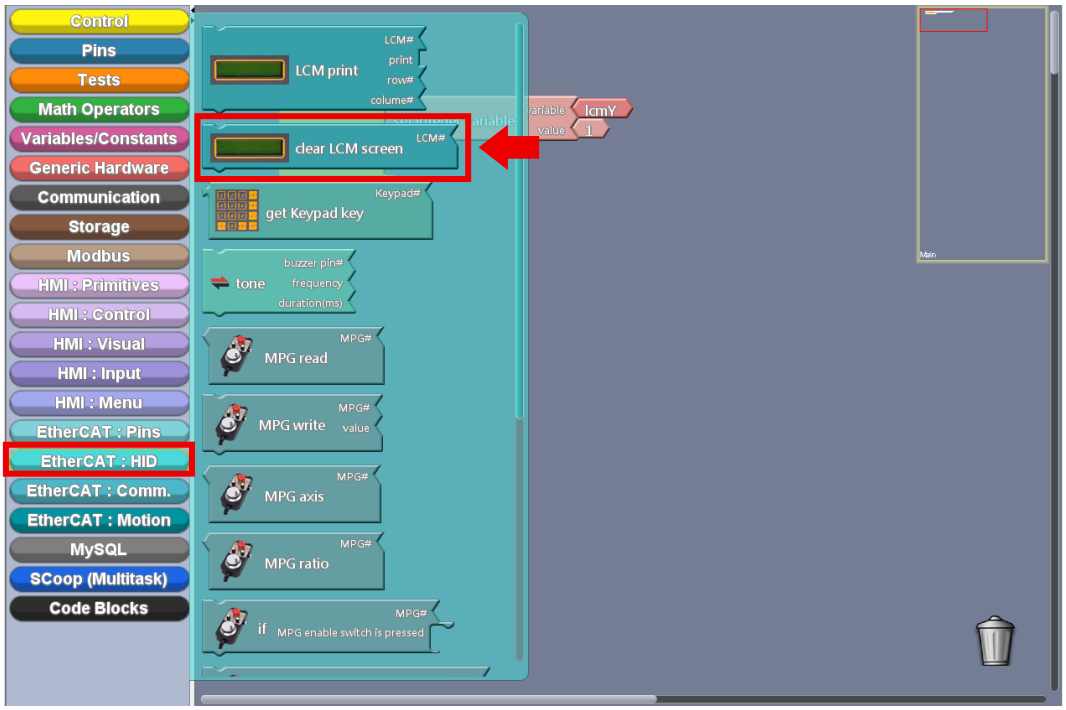


- # clears the LCM and sets the print row to **Row 1**.
- * clears the LCM and sets the print row to **Row 2**.
- Keys 0–9 print at columns **1–10**; keys A–D print at columns **11–14** of the current row.

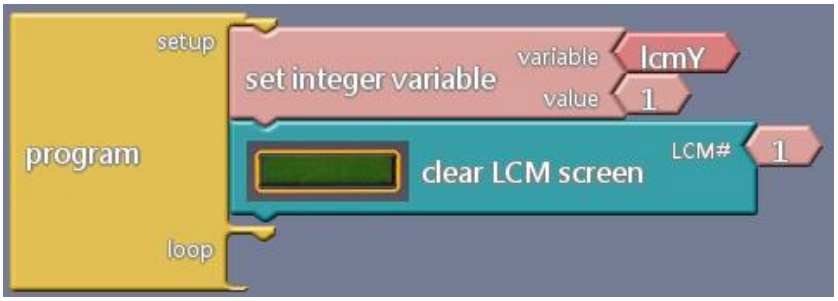
First, in the "**setup**" area in the "**program**" block, we use the "**set integer variable**" block from the "**Variables/Constants**" class to create a value to record the row number.

And change the variable name to "**lcmY**", and value to 1.

We use the “**clear LCM screen**” block from “**EtherCAT : HID**” class, to initialize the LCM through “**Virtual Lcm1**” (we select “**Virtual Lcm1**” for HU9S LCM).

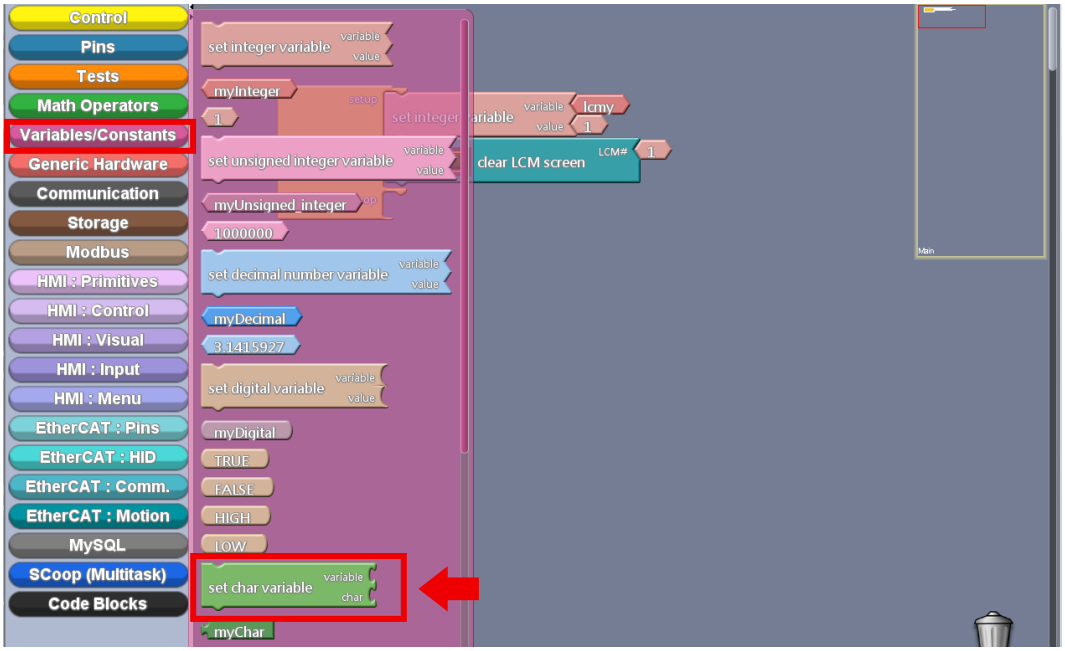


Like this.

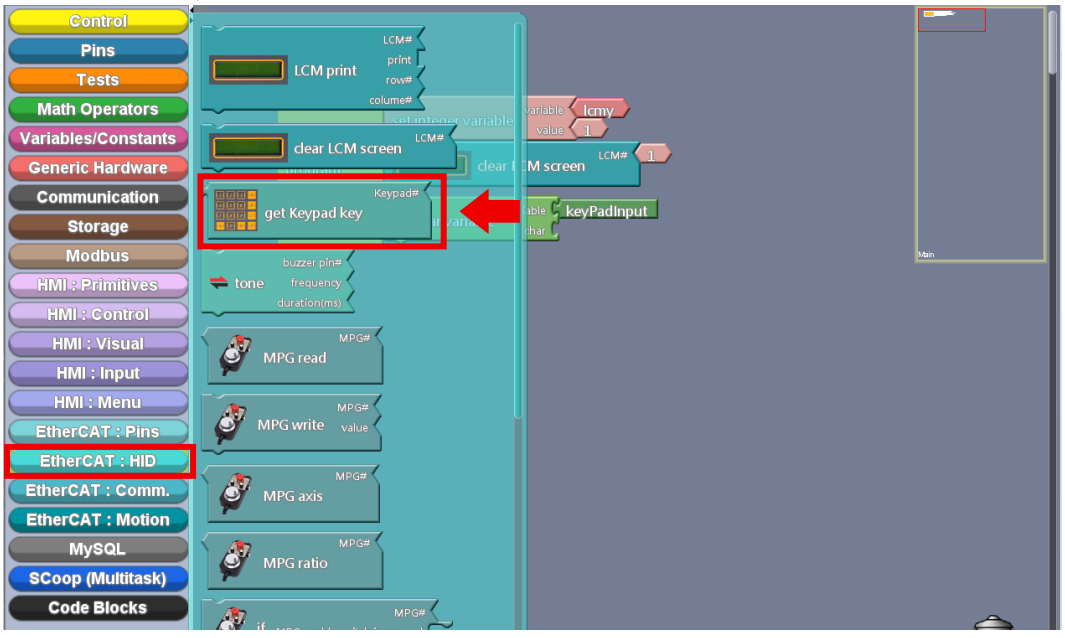


Next, we move on the “**loop**” area in the “**program**” block.

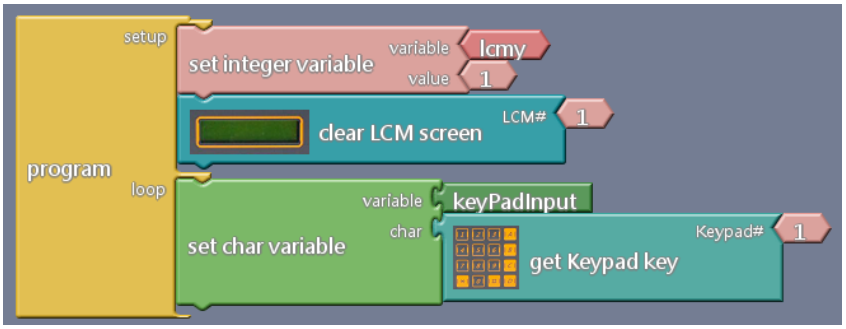
We use the “**set char variable**” block from the “**Variables/Constants**” class to create a value to record the keypad input value.



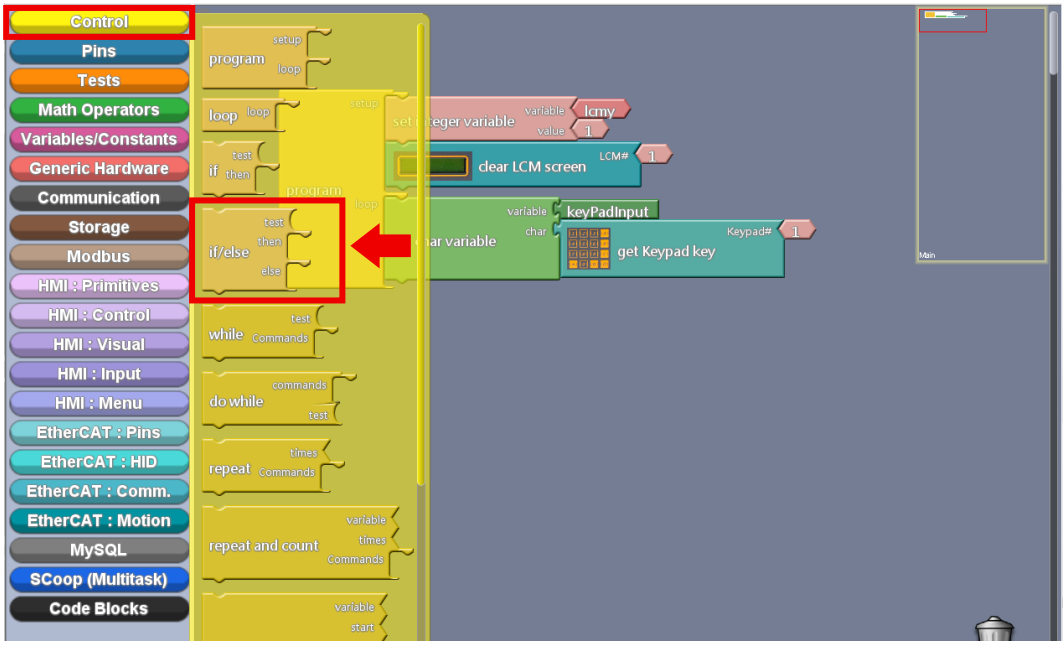
Change the variable name to “**keyPadInput**”. And we use the “**get Keypad key**” block from the “**EtherCAT : HID**” class for the “**char**” area of the “**keyPadInput**”. (we select “**Virtual Keypad1**” for HU9S Keypad).



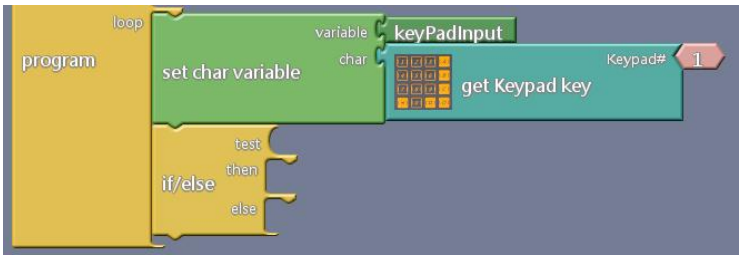
Like this.



We use the "**if/else** " block from the "**Control**" class to the program's main loop to determine whether the keypad input value.

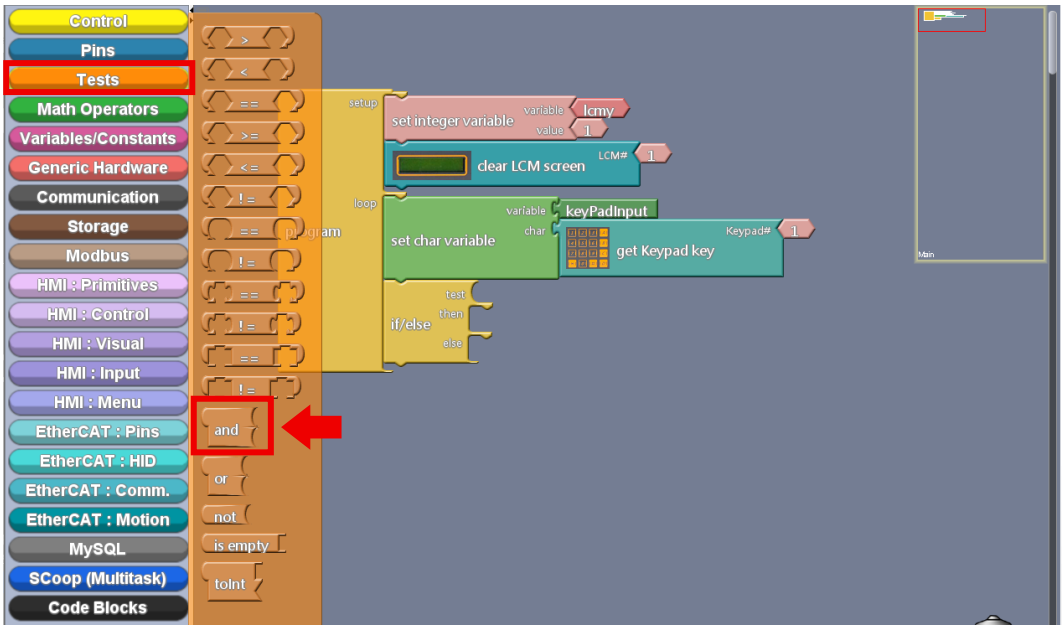


Like this.



First, we want to determine the keypad input value Digits '0'..'9'. Put the row = lcmY, and column = key - '0' + 1 ('0'→1 … '9'→10).

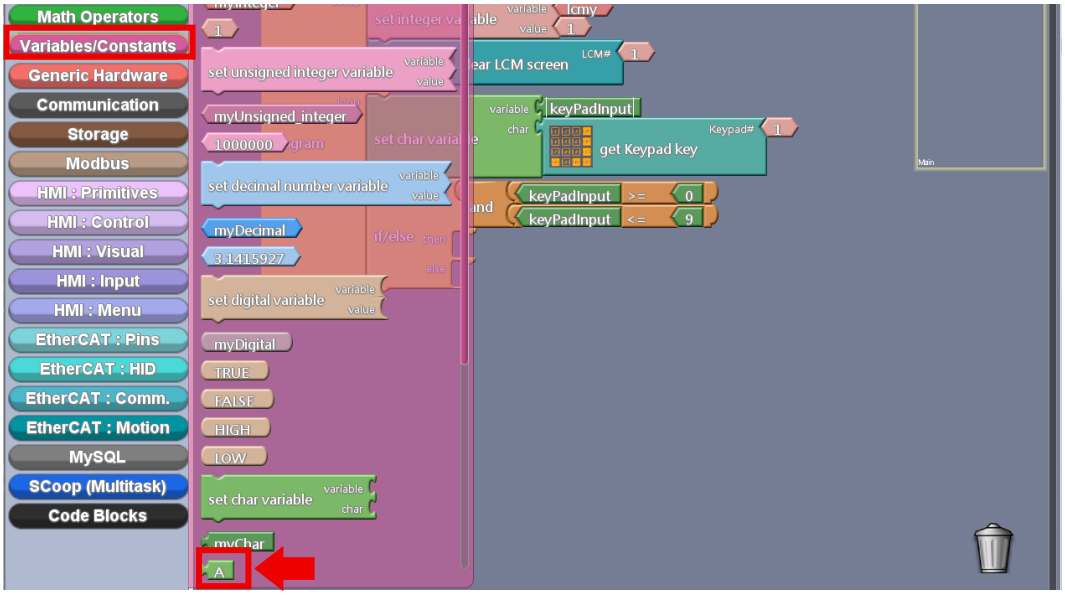
We use the "**and**" block from the "**Tests**" class to the "**test**" area of the "**if/else**" block.



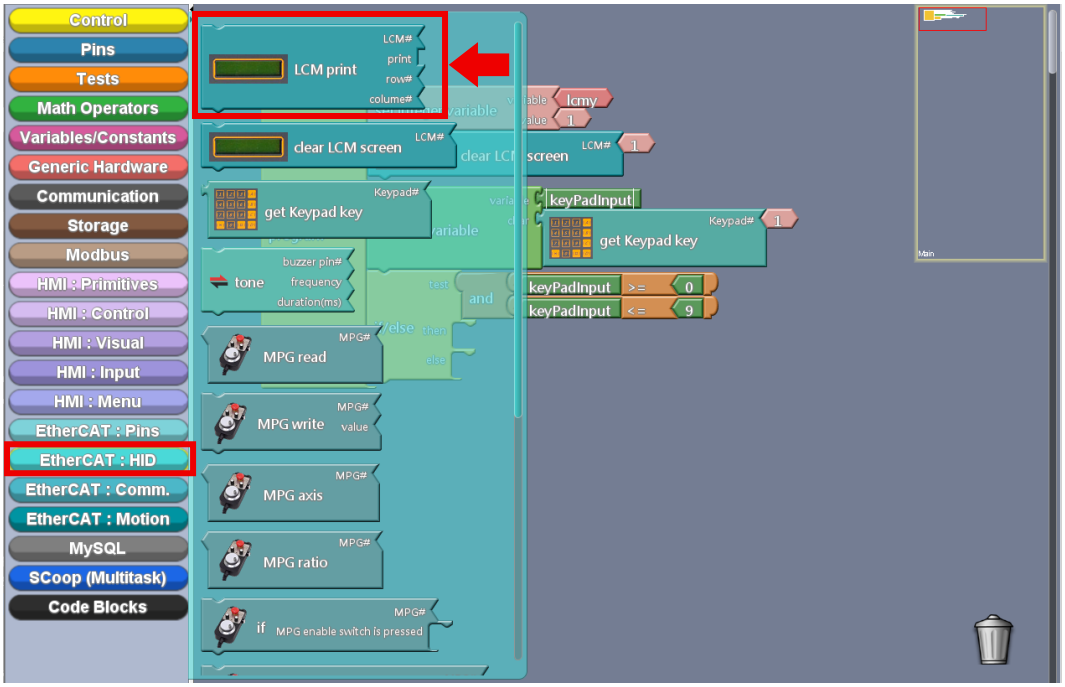
And we use the "**>=**" and the "**<=**" blocks to determine the "**keyPadInput**" value between **'0'** to **'9'**.

Like this.

*Note the "**keyPadInput**", "**0**", "**9**" blocks are all "**char**" variable in the "**Variables/Constants**" class.

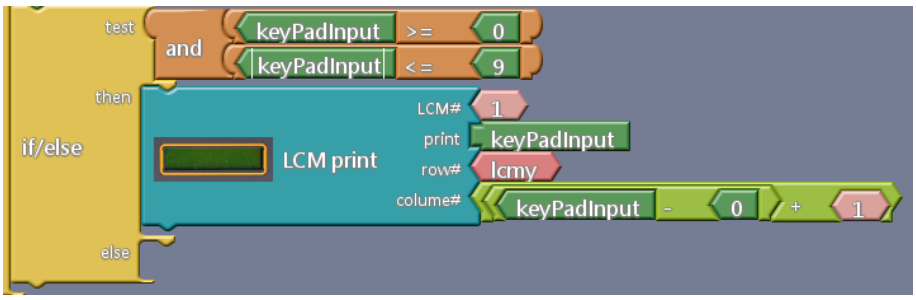


We use the “**LCM print**” block from the “**EtherCAT : HID**” class to the “**then**” area of the “**if/else**” block.

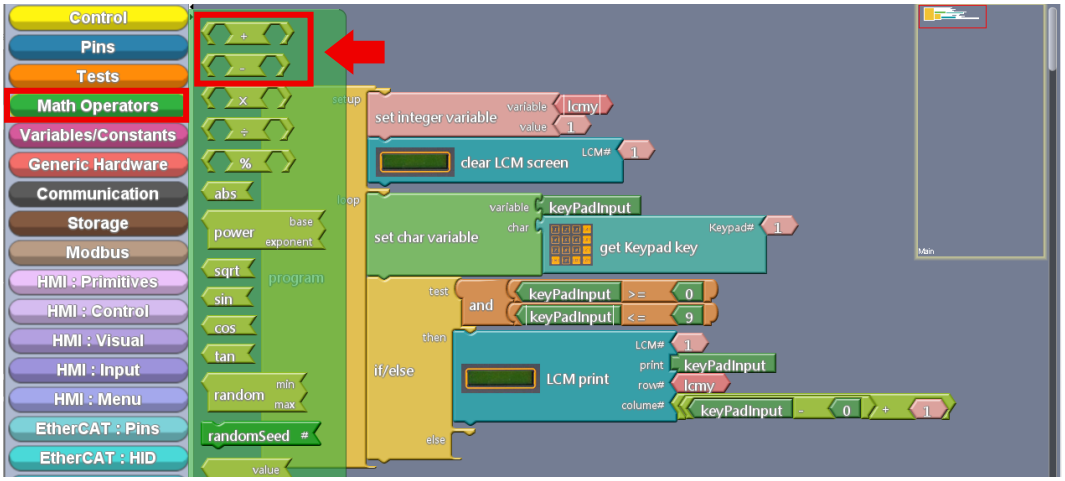


And we use the “**Virtual Lcm1**”, print out the “**keyPadInput**” value, row = “**lcmY**”, and column = **“KeyPadInput” - '0' + 1**.

Like this.

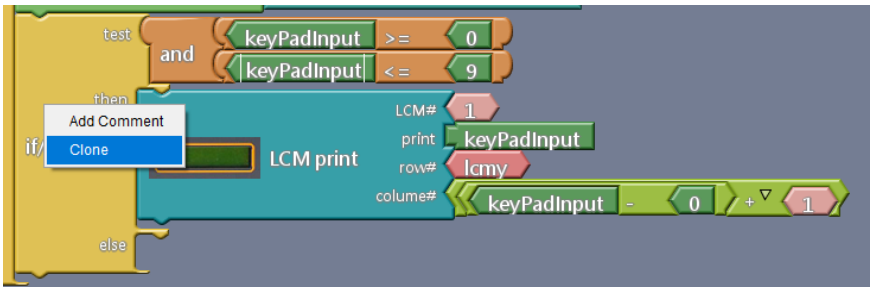


*__Note__ the math **“+/-“** blocks are in the “**Math Operators**” class.



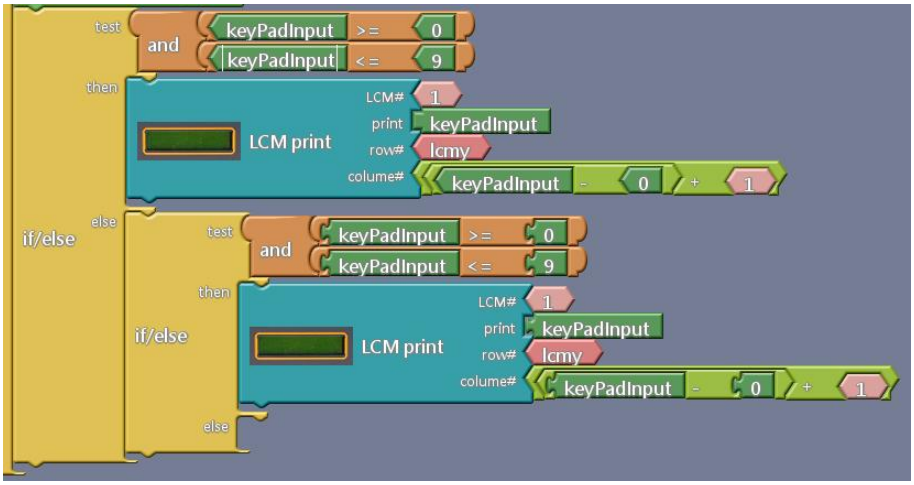
Next, in the “**else**” area of the “**if/else**” blocks, we want to determine the keypad input value Letters 'A'..'D'. Put the row = lcmY, and column = key - 'A' + 11 (11..14).

We click the right button on the “**if/else**” blocks, and it’ll show two options: “**Add Comment**” and “**Clone**”.



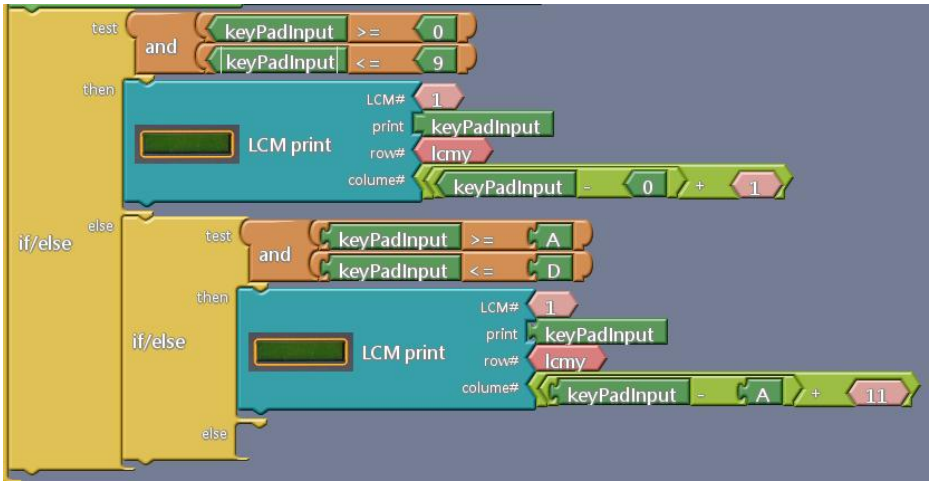
We click “**Clone**” to duplicate all the blocks in the "**if/else**" we set up and adjust their parameters to what we need. And put the duplicate “**if/else**” blocks under the “**else**” area of the “**if/else**” blocks.

Like this.



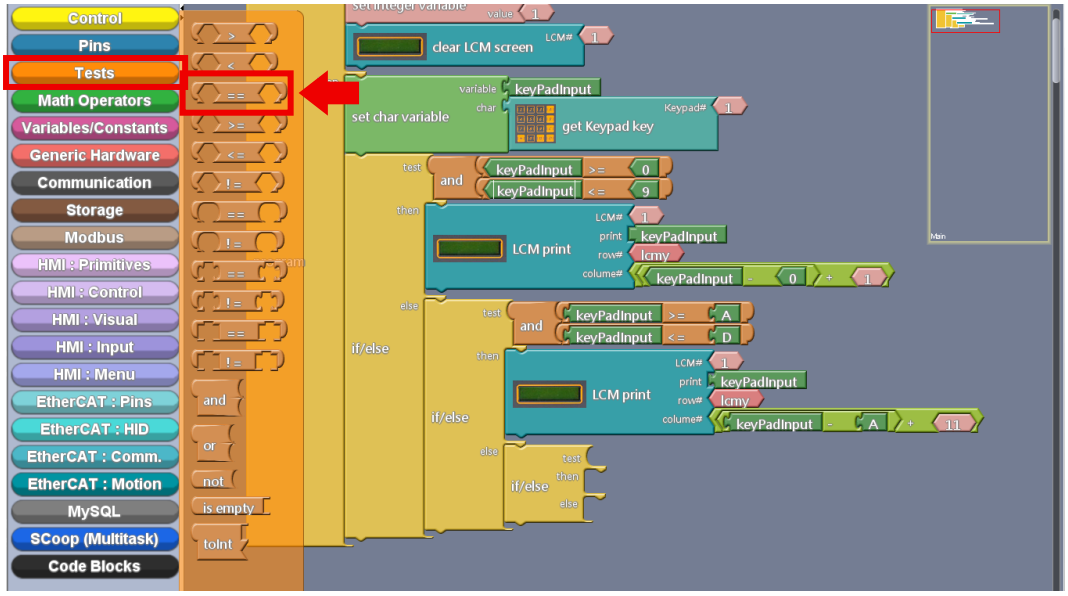
We change the determining conditions from the “**keyPadInput**” value between ‘**0**’ and ‘**9**’ to ‘**A**’ and ‘**D**’ in the “**and**” block of the “**test**” area.
Next, we change the “**column**” value from **“KeyPadInput” - '0' + 1** to **“KeyPadInput” - 'A' + 11**.

Like this.



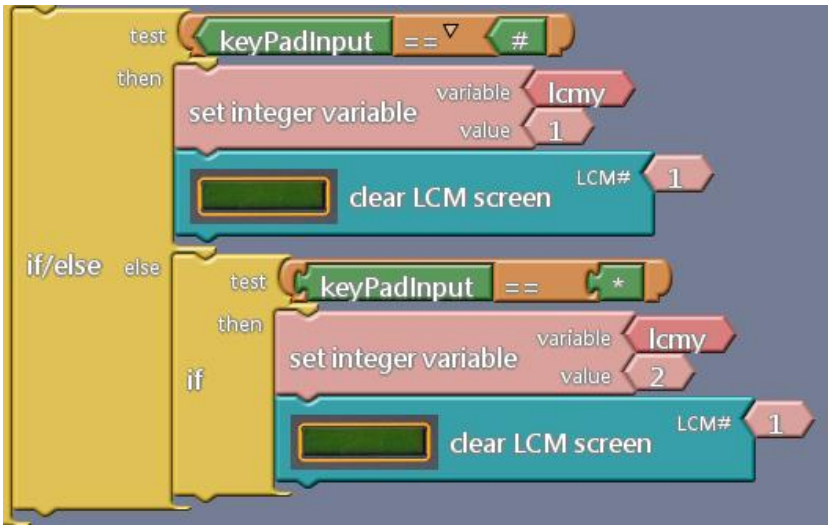
We create another “**if/else**” block to determine the keypad input value “**#**” and “*****”. This block is under the “**else**” area of the “**if/else**” block, which determines the ‘A’ to ‘D’.

We use the “**==**” block in the “**Test**” class to determine whether the “**keyPadInput**” value is “**#**” or “*****”.



First, in the “**test**” area of the “**if/else**” block, we determine if the “**keyPadInput**” value is “**#**”, and in the “**then**” area, we change the “**lcmY**” to 1 and use the “**clear LCM screen**” block to clear the LCM. Next, in the “**else**” area, we add another “**if**” block to check if the “**keyPadInput**” value is “*****”. If so, we set “**lcmY**” to 2 and use the “**clear LCM screen**” block to clear the LCM.

Like this.

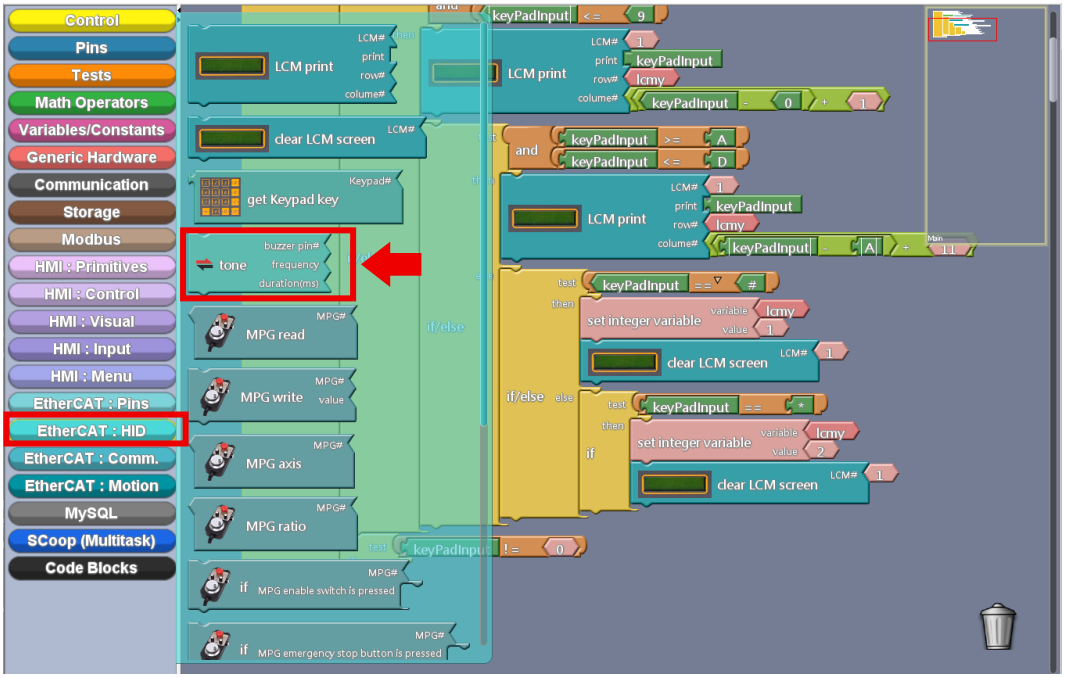


Finally, we want to trigger the buzzer at a frequency of 3000 Hz for 200 ms only when the key is pressed.

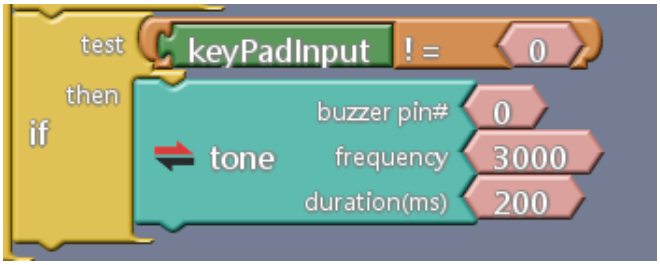
We use the "**if** " block from the "**Control**" class in the program's main loop to determine whether the keypad input value is pressed.
And in the "**test**" area of the "**if**" block, we use the "**!=**" block to determine whether the "**keyPadInput**" value is not 0.

Like this.

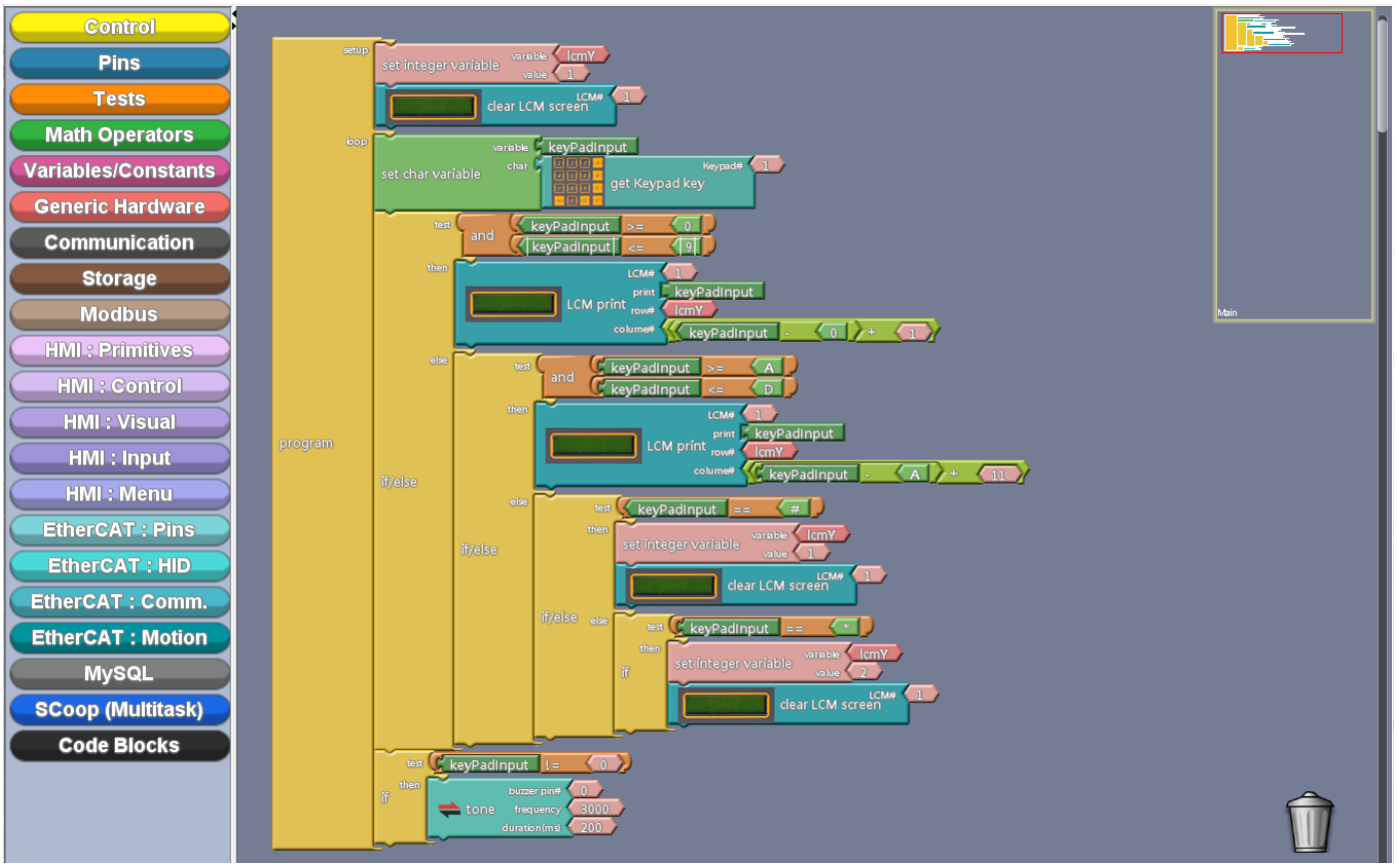
Next, we use "**tone**" block in the "**EtherCAT : HID**" class to the "**then**" area of the "**if/else**" block.



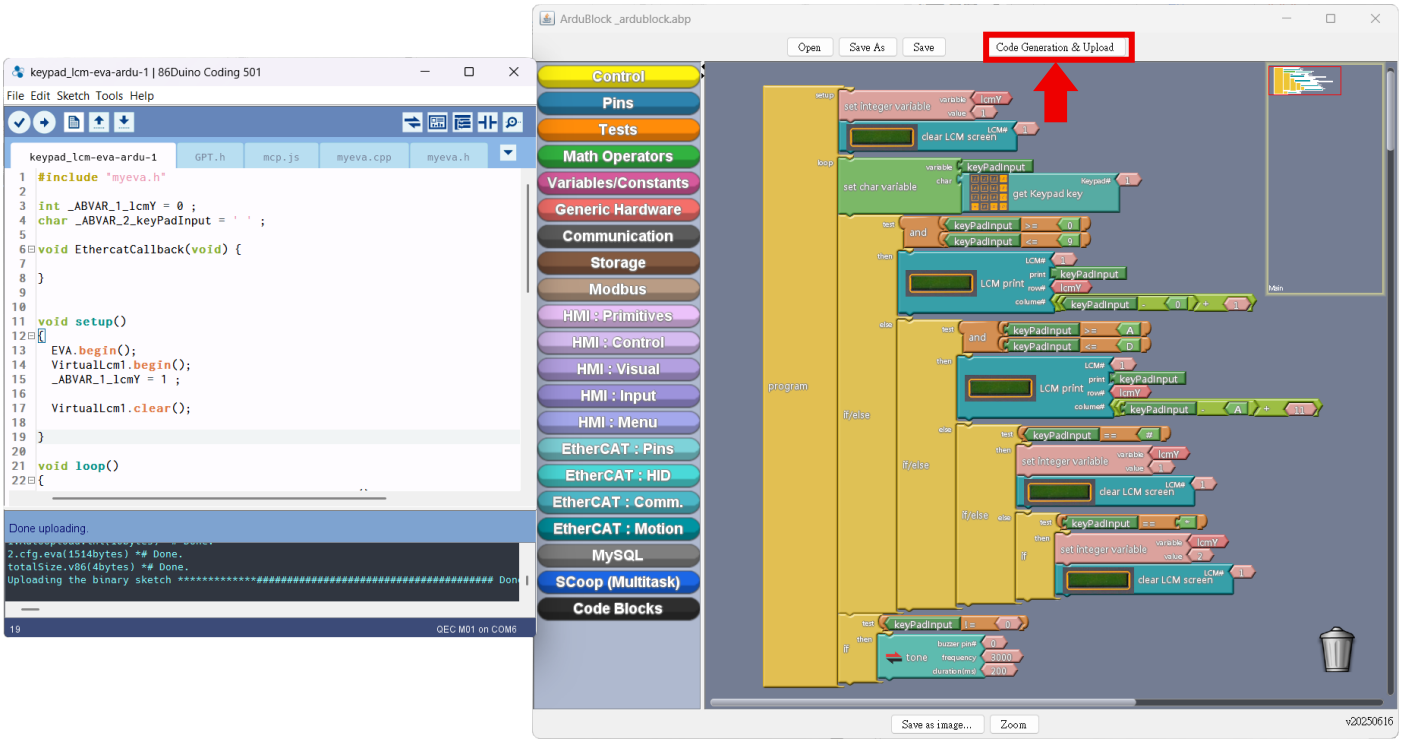
We set the tone buzzer at a frequency of 3000 Hz for 200 ms
Like this.



This is the full block.



After you finish, click the “**Code Generation & Upload**” button, and ArduBlock will automatically generate the program code and upload it to the QEC-M-01.

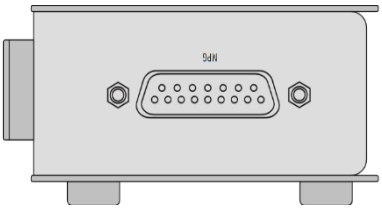


After uploading, you can press the keypad on the QEC-R11HU9S-N's side and printing on the specific positions of LCM according to it. Buzzer will buzz when the keypad is pressed.
Among them:

- # clears the LCM and sets the print row to **Row 1**.
- * clears the LCM and sets the print row to **Row 2**.
- Keys 0–9 print at columns **1–10**; keys A–D print at columns **11–14** of the current row.

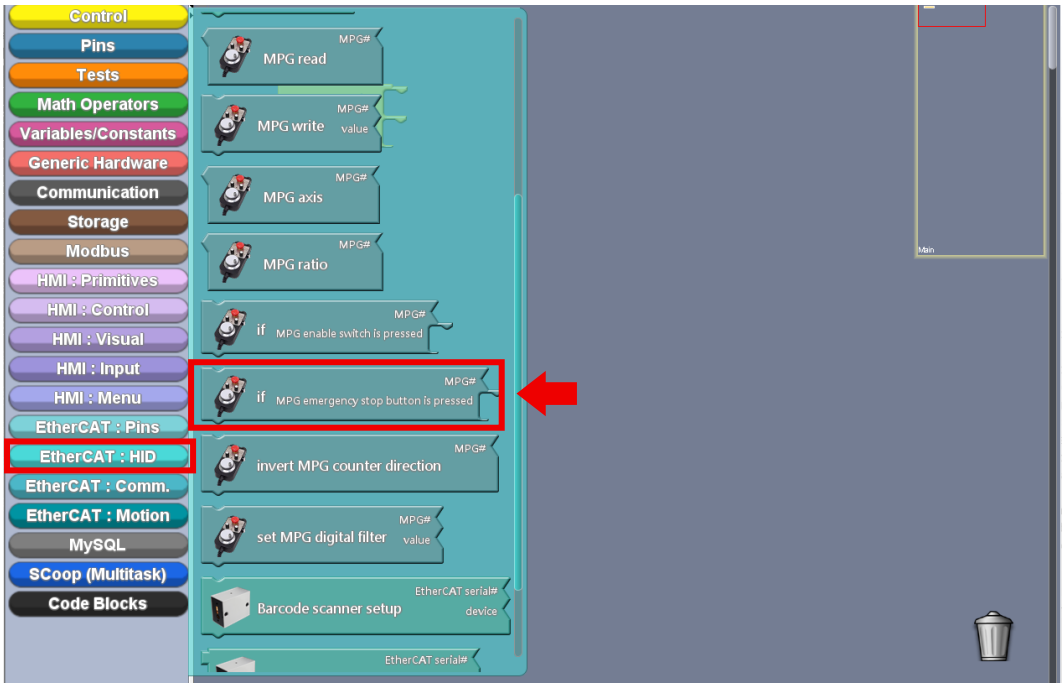
Step 4.3: MPG Hand wheel

In this section, we will read the MPG data and status for QEC-R11HU9S and print EMG, Enable, Axis, Ratio, and Raw data through the Serial Monitor of the 86Duino IDE.

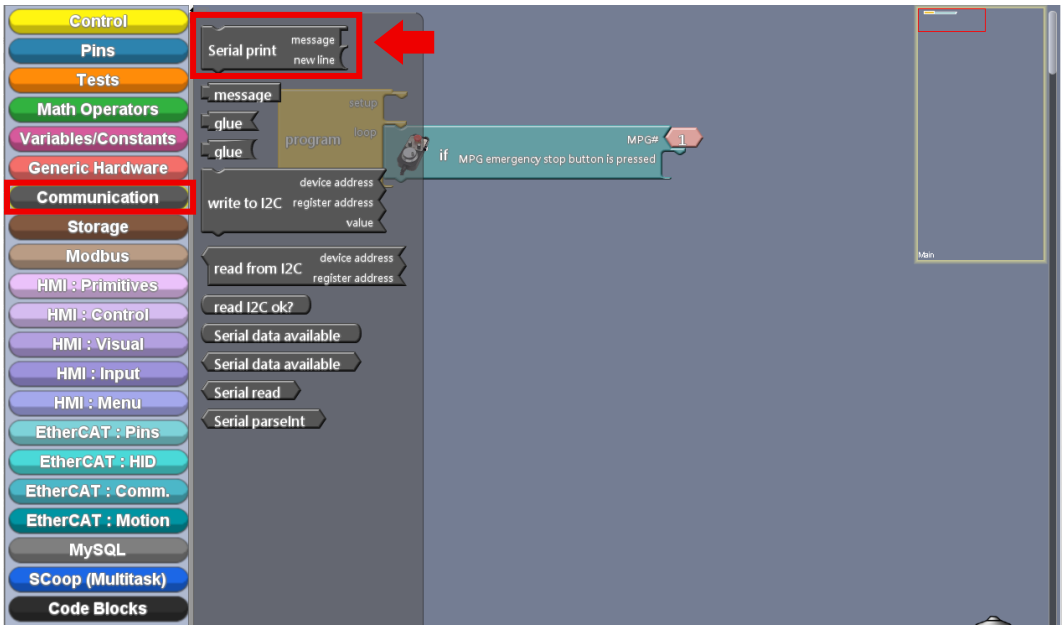


No.	Pin Assignment	No.	Pin Assignment
1	VCC	9	AXIS_B0
2	A	10	AXIS_B1
3	A-	11	AXIS_B2
4	B	12	MULTIPLE_B0
5	B-	13	MULTIPLE_B1
6	C	14	EMERGENCY
7	C-	15	LED
8	GND	-	-

First, we use the “**if MPG emergency stop button is pressed**” block in the “**EtherCAT : HID**” class in the “**loop**” area of the “**program**” block. (we select “**Virtual MPG1**” for HU9S MPG).

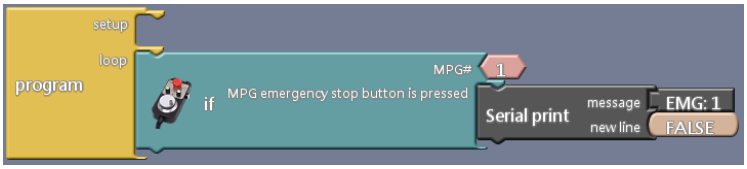


And we use the “**Serial print**” block in the “**Communication**” class, and put it in the “**if MPG emergency stop button is pressed**”.

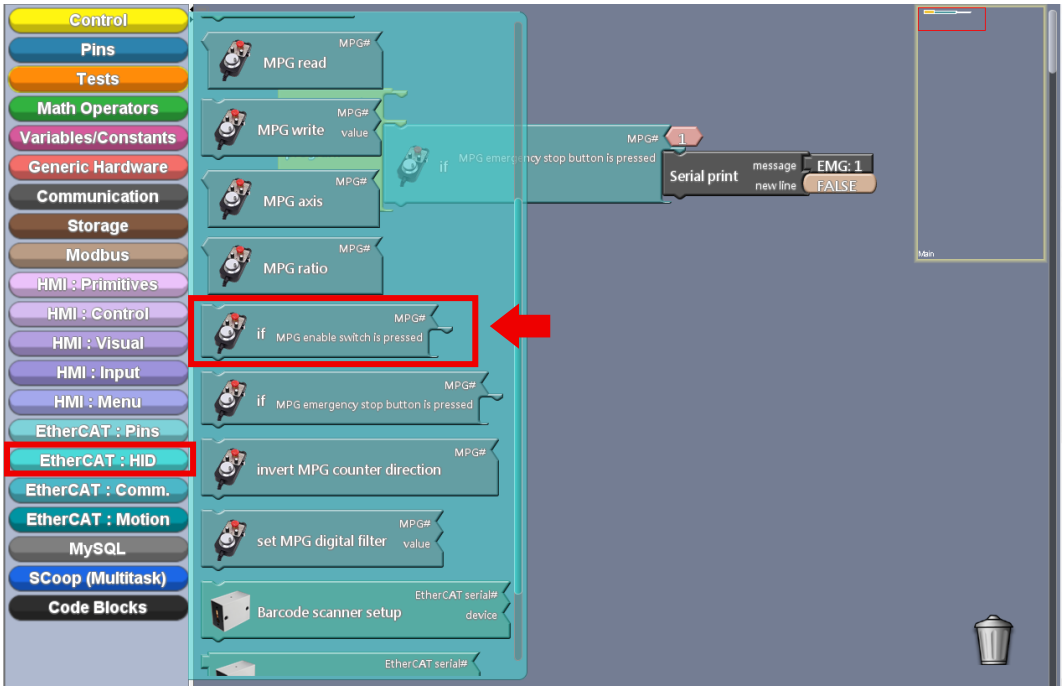


We set the “**message**” to “**EMG: 1**”, and “**new line**” to “**FALSE**” in the “**Serial print**” block.

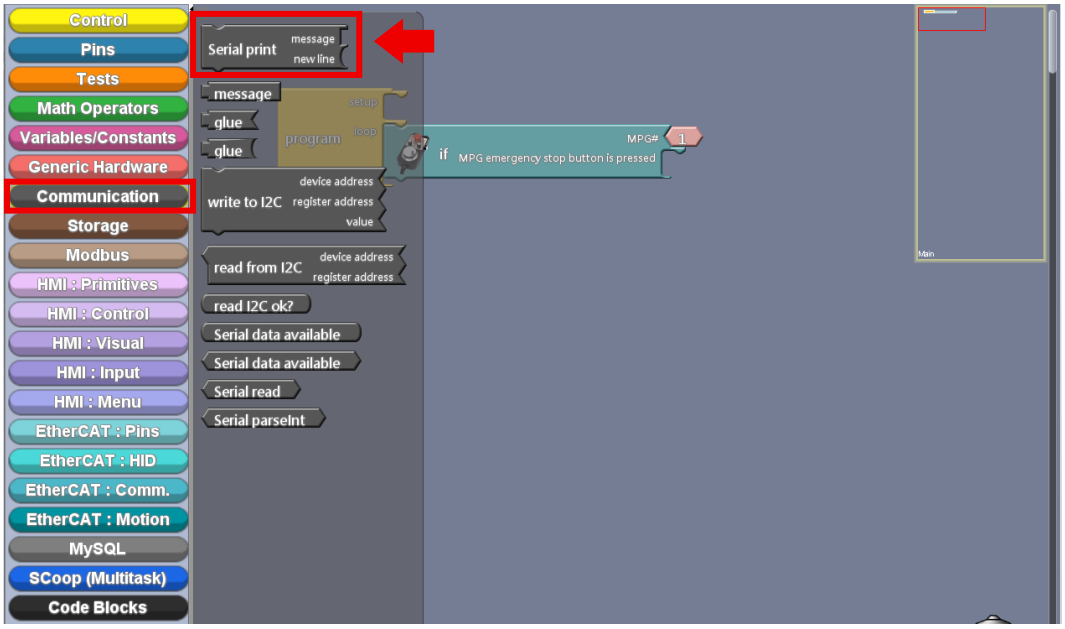
Like this.



Next, we use the “**if MPG enable switch is pressed**” block in the “**EtherCAT : HID**” class.

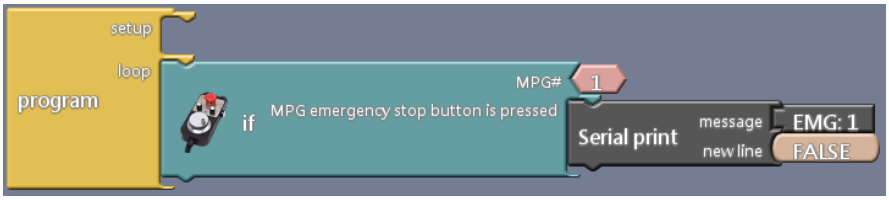


Same as above, we use the “**Serial print**” block in the “**Communication**” class, and put it in the “**if MPG enable switch is pressed**”.



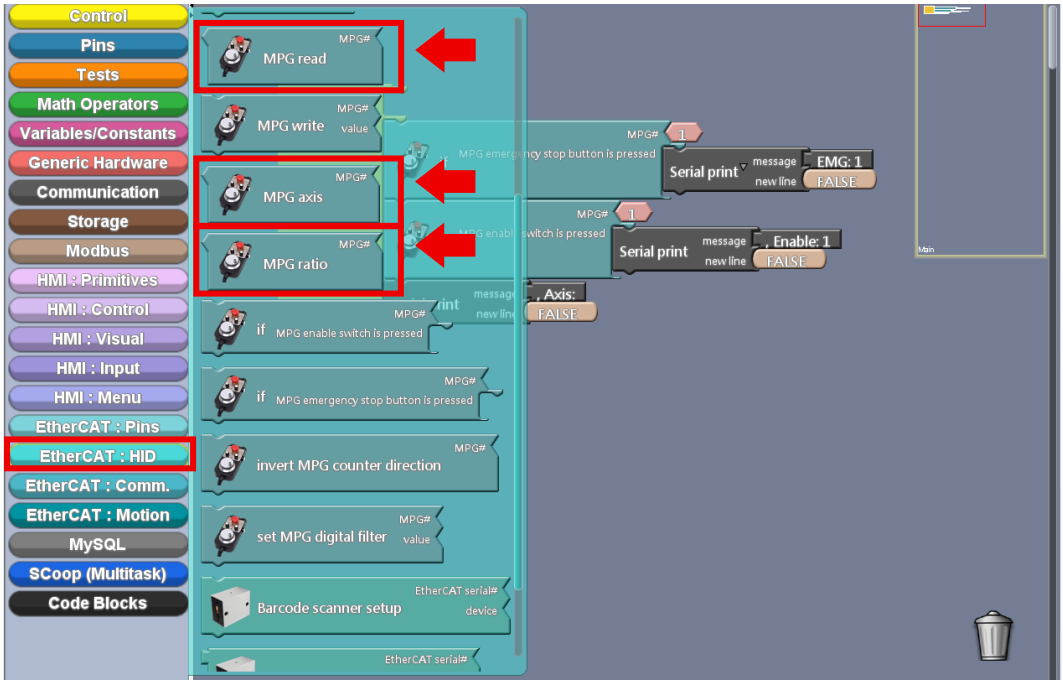
We set the “**message**” to “**, Enable: 1**”, and “**new line**” to “**FALSE**” in the “**Serial print**” block.

Like this.

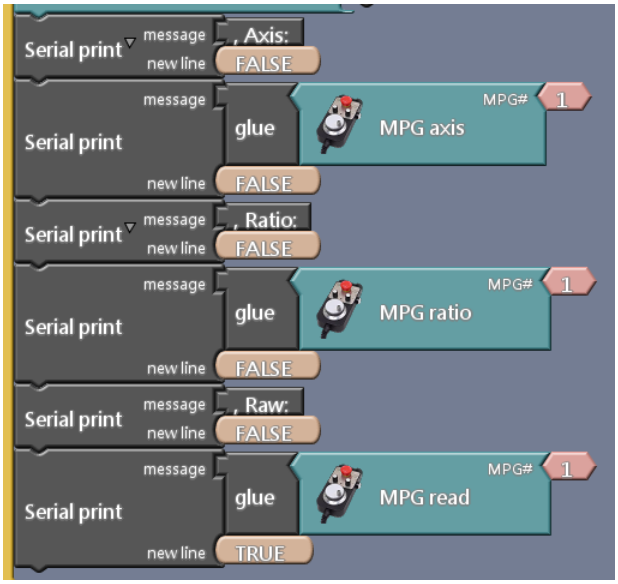


Then, we print out the axis, ratio, and raw data.

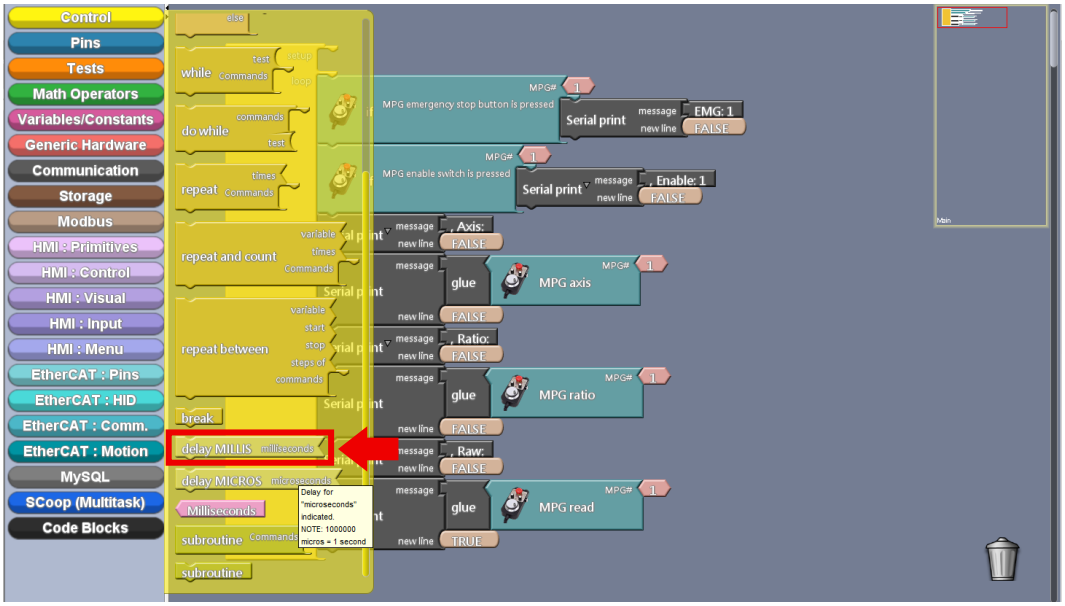
We use three “**Serial print**” blocks and set the “**message**” to “**, Axis:**”, “**, Ratio:**”, and “**, Raw:**”.
And we use the “**MPG axis**” block, the “**MPG ratio**” block, and the “**MPG read**” block in the “**EtherCAT : HID**” class.



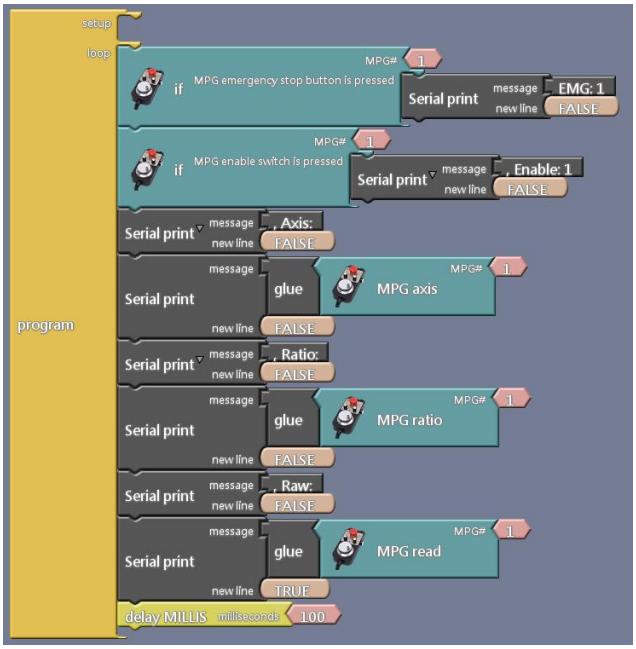
We put the "**Serial print**" block and the "**EtherCAT : HID**" block in the "**loop**" area of the "**program**" block. And we need to change the message content to the "**EtherCAT : HID**" block.
We use the “**glue**” block from the “**Communication**” Class to glue the "**EtherCAT : HID**" block to the "**message**" area of the “**Serial Print**” block.
Like this.



Next, because we need to wait for the MPG response time, we put a "**delay MILLIS**" block after all blocks.



We set the 100 ms to the "**delay MILLIS**" block.
Like this.

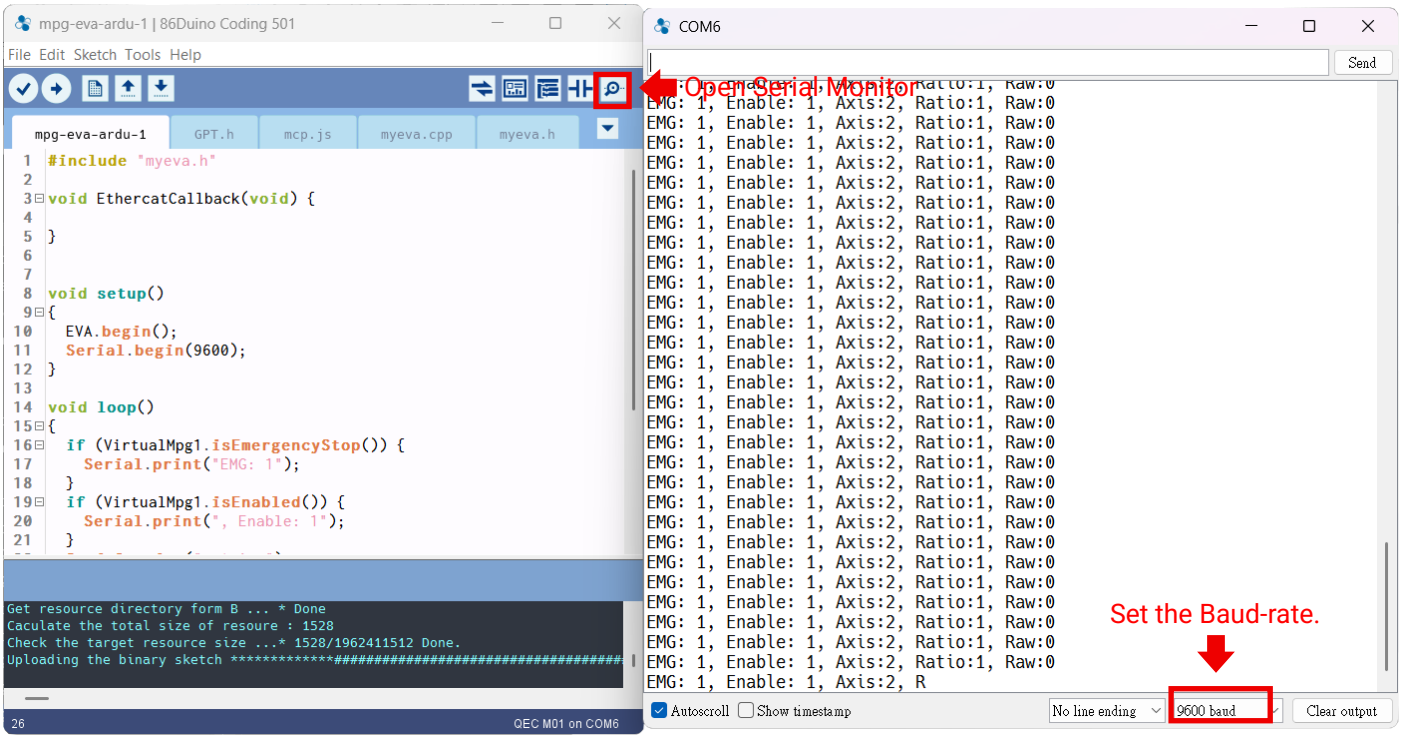


***Note:**

The default baud-rate of Serial monitor is 9600.

After you finish, click the “**Code Generation & Upload**” button, and ArduBlock will automatically generate the program code and upload it to the QEC-M-01.

After the upload is completed, you can read the data and status of the MPG of QEC-R11HU9S, and view EMG, Enable, Axis, Ratio, and RAW data through the Serial Monitor of 86Duino IDE.



Troubleshooting

QEC-M-01 cannot successfully upload code

When you are unable to successfully upload code, please open 86EVA to check if your QEC EtherCAT MDevice's environment is abnormal. As shown in the figure below, please try updating your QEC EtherCAT MDevice's environment, which will include the following three items: Bootloader, EtherCAT firmware, and EtherCAT tool.

Now, we will further explain how to proceed with the update:

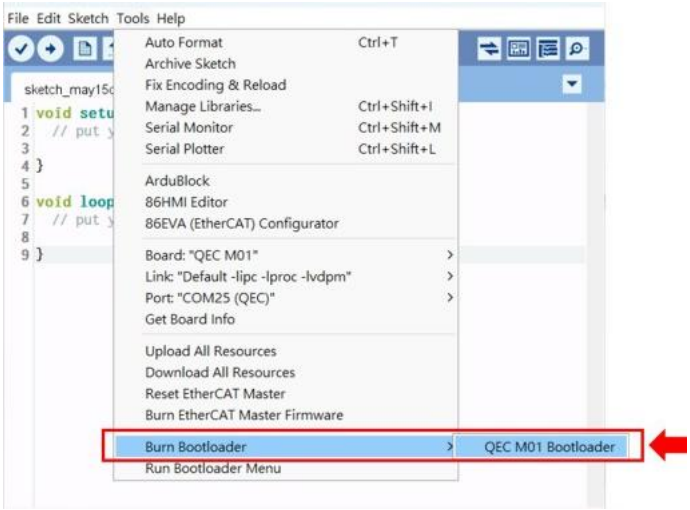
Step 1: Setting up QEC-M

1. Download and install 86Duino IDE 500+ (or a newer version).
   You can download it from Software.
2. Connect the QEC-M: Use a USB cable to connect the QEC-M to your computer.
3. Open 86Duino IDE: After the installation is complete, open the 86Duino IDE software.
4. Select Board: From the IDE menu, choose "**Tools**" > "**Board**" > "**QEC-M-01**" (or the specific model of QEC-M you are using).
5. Select Port: From the IDE menu, choose "**Tools**" > "**Port**" and select the USB port to which the QEC-M is connected.

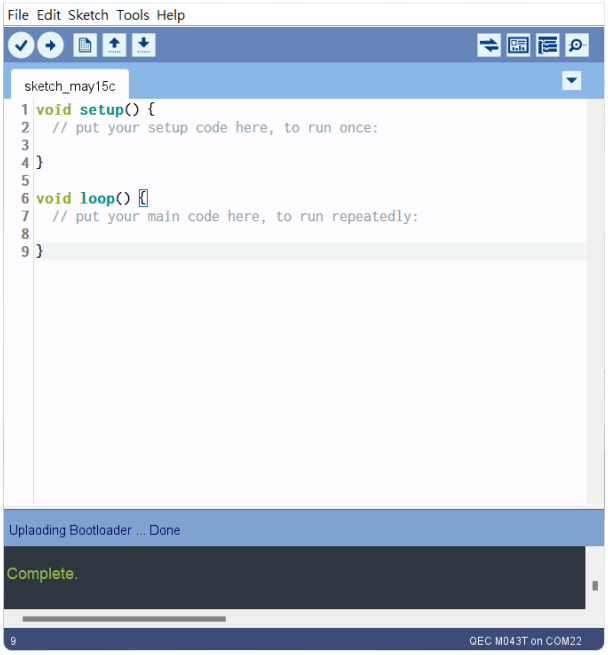
Step 2: Click “Burn Bootloader” button

After connecting to your QEC-M product, go to “**Tools**”> “**Burn Bootloader**”.
The currently selected QEC-M name will appear. Clicking on it will start the update process, which will take approximately 5-20 minutes.

- QEC-M-01:



Step 3: Complete the Update



After completing the above steps, your QEC-M has been successfully updated to the latest version of the development environment.

Warranty

This product is warranted to be in good working order for a period of one year from the date of purchase. Should this product fail to be in good working order at any time during this period, we will, at our option, replace or repair it at no additional charge except as set forth in the following terms. This warranty does not apply to products damaged by misuse, modifications, accident or disaster. Vendor assumes no liability for any damages, lost profits, lost savings or any other incidental or consequential damage resulting from the use, misuse of, originality to use this product. Vendor will not be liable for any claim made by any other related party. Return authorization must be obtained from the vendor before returned merchandise will be accepted. Authorization can be obtained by calling or faxing the vendor and requesting a Return Merchandise Authorization (RMA) number. Returned goods should always be accompanied by a clear problem description.