Blink Program

Obtaining and Setting up Software

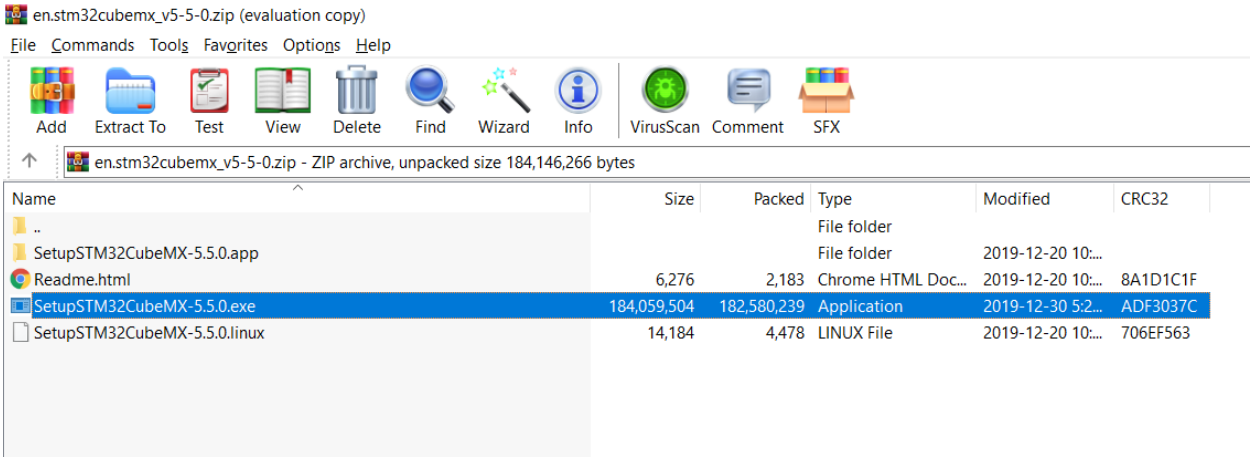
To be able to interact with the MCU the software STM32 and Atollic are required.

Atollic

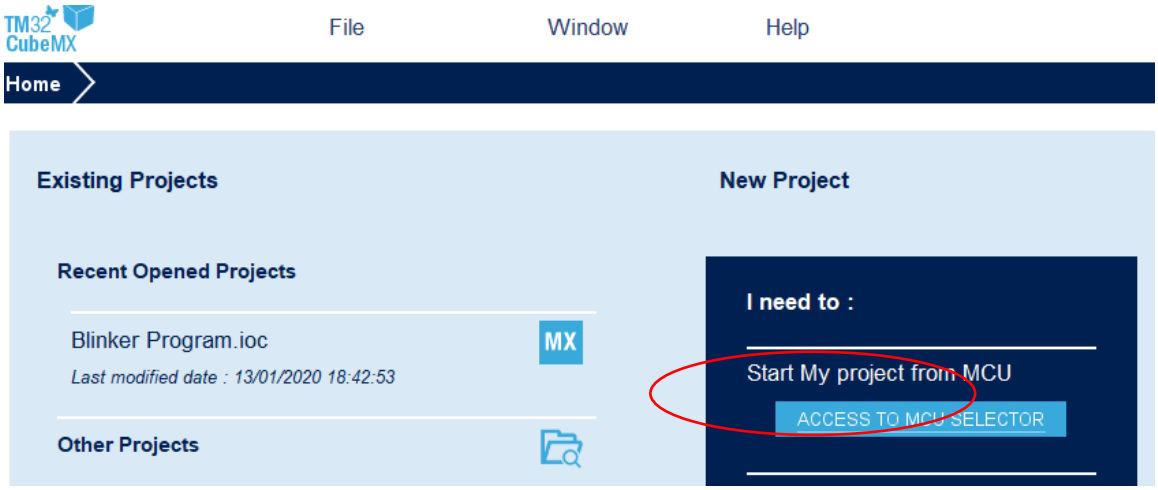
Atollic will be the IDE used in this project which can be obtained from this link. Follow all the sign-up requirements to download the software and nothing else should be necessary.

STM32

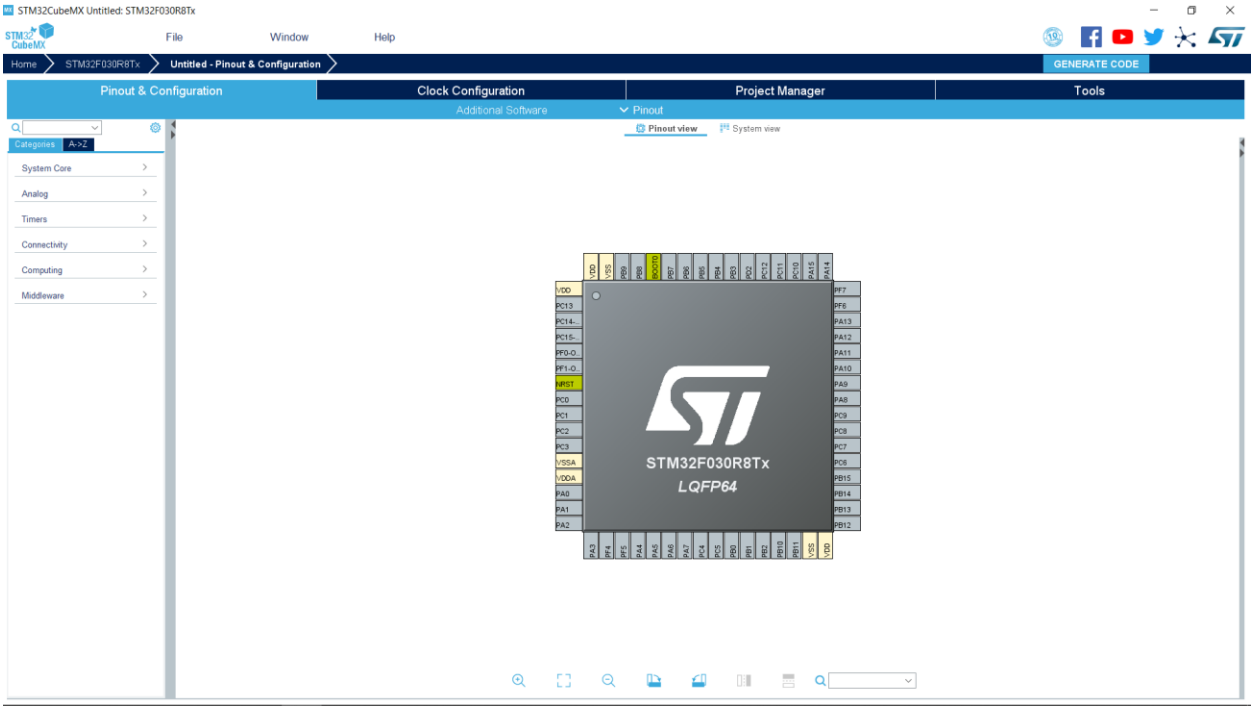
STM32 is the piece of software that configures the pins to on an MCU. To download this software visit this link. The download should be a zipped file containing the program “SetupSTM32CubeMX-5.5.0.exe”. Run the program and follow all setup prompts.



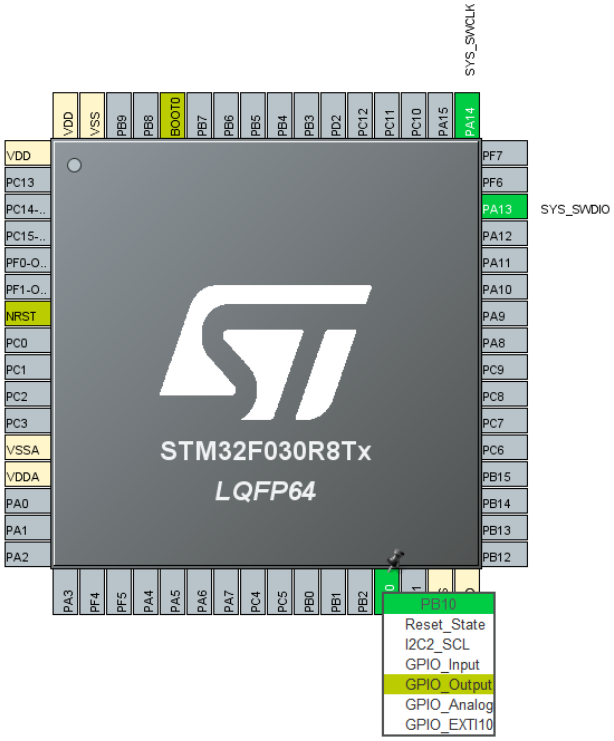
Open the program and a screen like the one below should appear. From here select “Access to MCU selector” and a list of MCU’s will open.



From this list look under the reference column for “STM32F030R8Tx” and “LQFP64” under the package column. Once the right MCU is found, click “start project” at the top right. If everything was done right, the screen below should appear.



Before anything else is done click on the “System Core” tab, and then the “SYS” option. Check off the “Debug Serial Wire” box and the click the drop-down menu for the “Timebase Source” and click “TIM1.”. After this the pins are ready to be configured.



Each pin has its own unique, location and name dedicated to itself. These names can be matched up with the physical MCU itself. The name of each pin should be available if the mouse is hovered over the pins. Each of these pins can all have specific function. For example, VDD provides the voltage to the chip and VSS provides the ground. To modify the function of a pin click it and a drop-down menu will appear. For the purposes of the blink program, click “GPIO_Output” option on pin 29. Note: This can be configured to be any pin but for it to match up with further steps picking pin 29 will make things easier. After this, click on the Project Manager tab found at the top.

"Project Name" will allows a title to be given to the project and Project Location" controls where on the computer the generated code to be saved. The "Toolchain / IDE" drop down menu will allow you to specify which IDE the code will be generated. In this case the "TrueSTUDIO" option will be selected. Now click "GENERATE CODE" at the top right and say accept any prompts for further downloads. The program should generate a folder containing code based on the specifications made. Open the file made and open ". project." Once in Atollic find the project in the project explorer and click the drop down, and then click the drop down on "Src" where "main.c" can be seen. Double click on the "main.c" and the code should be visible.

Programming

With the code now visible, it is time to edit it. For this specific program only 2 functions will be added, HAL_GPIO_TogglePin(), and HAL_Delay().

HAL_GPIO_TogglePin

Function name	**void HAL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)**
Function description	Toggles the specified GPIO pins.
Parameters	• **GPIOx:** Where x can be (A..K) to select the GPIO peripheral for STM32F429X device or x can be (A..I) to select the GPIO peripheral for STM32F40XX and STM32F427X devices. • **GPIO_Pin:** Specifies the pins to be toggled.
Return values	• **None:**

HAL_GPIO_TogglePin() will be used to toggle the voltage from 0V to 3.3V and as a result turning the LED on and off. The GPIOx is essentially the family of the pin, and the GPIO_Pin is the specific pin in the family. For this function to work the pin must be the pin selected to be the GPIO_Output pin during the STM32 setup. For example, if I wanted to toggle the pin (29)-PB10 my function would look like this: HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_10), where "B" is the family name and "10" is the specific pin in that family. If pin 29 was selected, the code in the example can be used verbatim. The general form for this would look like this HAL_GPIO_TogglePin(GPIO**"family"**, GPIO_PIN_**"Pin Number"**).

HAL_Delay

Function name	**void HAL_Delay (__IO uint32_t Delay)**
Function description	This function provides minimum delay (in milliseconds) based on variable incremented.
Parameters	• **Delay:** specifies the delay time length, in milliseconds.
Return values	• **None:**
Notes	• In the default implementation , SysTick timer is the source of time base. It is used to generate interrupts at regular time intervals where uwTick is incremented.

HAL_Delay() will be used to slow the toggling function down so the LED appears to be blinking to the naked eye. The argument just takes a number and delays the function by that number in milliseconds. For example, if the function were to be delayed by 100 milliseconds the function would appear as this: HAL_Delay(100).

For a continuous blinking, both these functions must be placed within the while loop in the generated code. This is found by scrolling down through the code. Place both functions between the “USER CODE BEING 3” and the “USER CODE END 3” comments so that if the code is regenerated in the future, the code will not get deleted in the process. The result should look like the picture below.

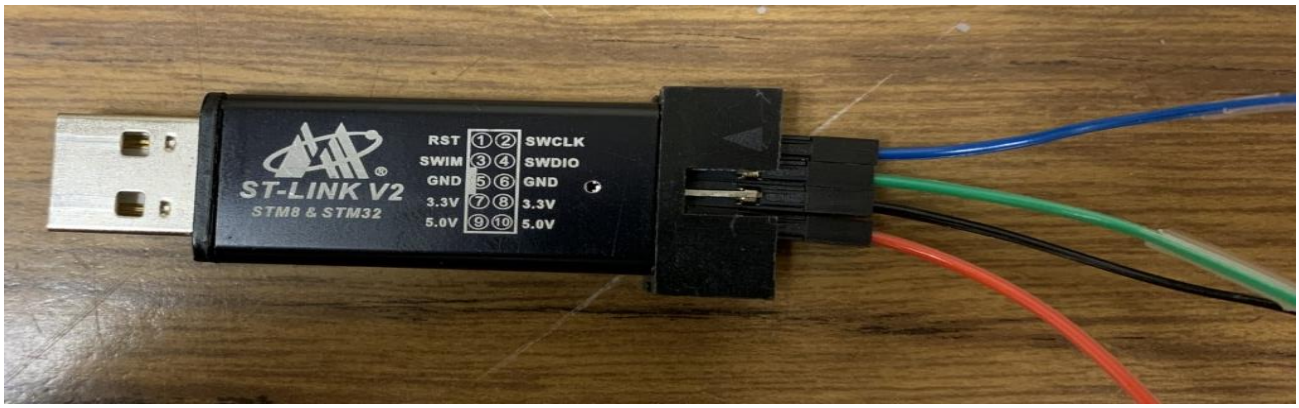
main.c

```
79   /* USER CODE END Init */
80
81   /* Configure the system clock */
82   SystemClock_Config();
83
84   /* USER CODE BEGIN SysInit */
85
86   /* USER CODE END SysInit */
87
88   /* Initialize all configured peripherals */
89   MX_GPIO_Init();
90   /* USER CODE BEGIN 2 */
91
92   /* USER CODE END 2 */
93
94
95
96   /* Infinite loop */
97   /* USER CODE BEGIN WHILE */
98   while (1)
99   {
100    /* USER CODE END WHILE */
101
102    /* USER CODE BEGIN 3 */
103      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_10);
104      HAL_Delay(100);
105
106
107  }
108  /* USER CODE END 3 */
109 }
110
111 /**
```

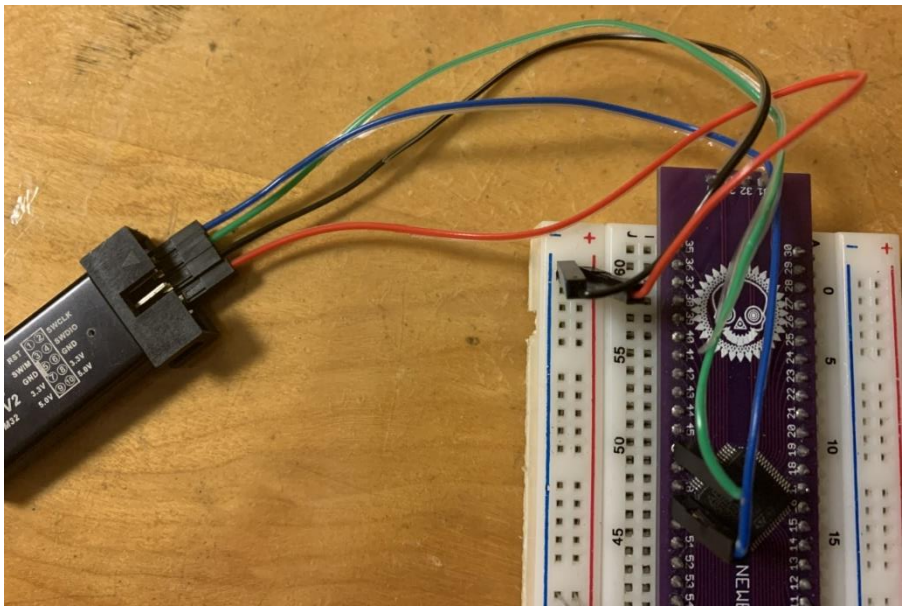
More Information on functions available in this IDE can be found here.

Wiring

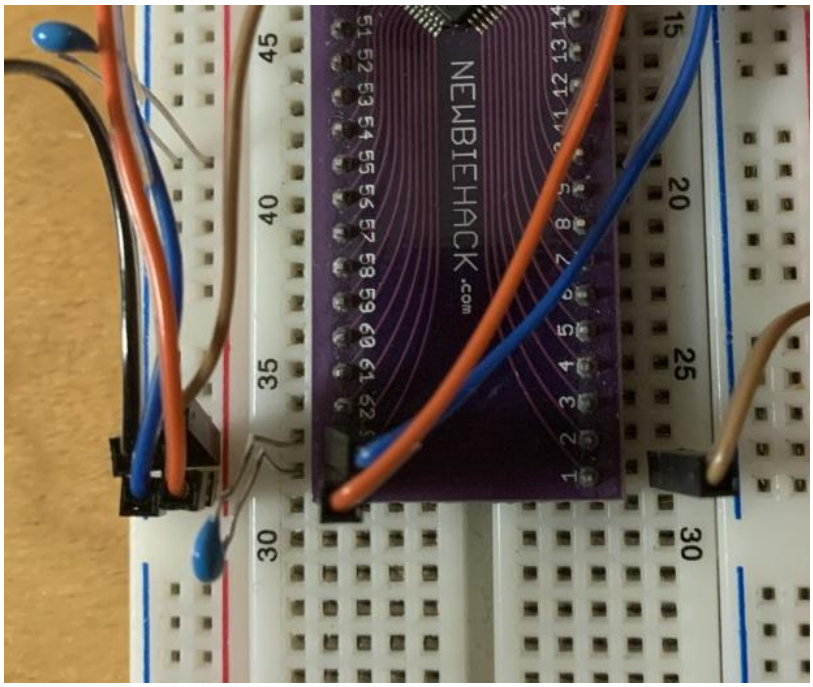
The components needed for the hardware portion of the project are an LED, two 0.1-0.2 microfarad capacitors, 9 jumper wires, a 390ohm resistor, a breadboard, an ST-Link, and the MCU. First, the ST-Link must be correctly wired. There should be 4 wires attached to the SWCLK, SWDI, GND, and 3.3V pins on the device.



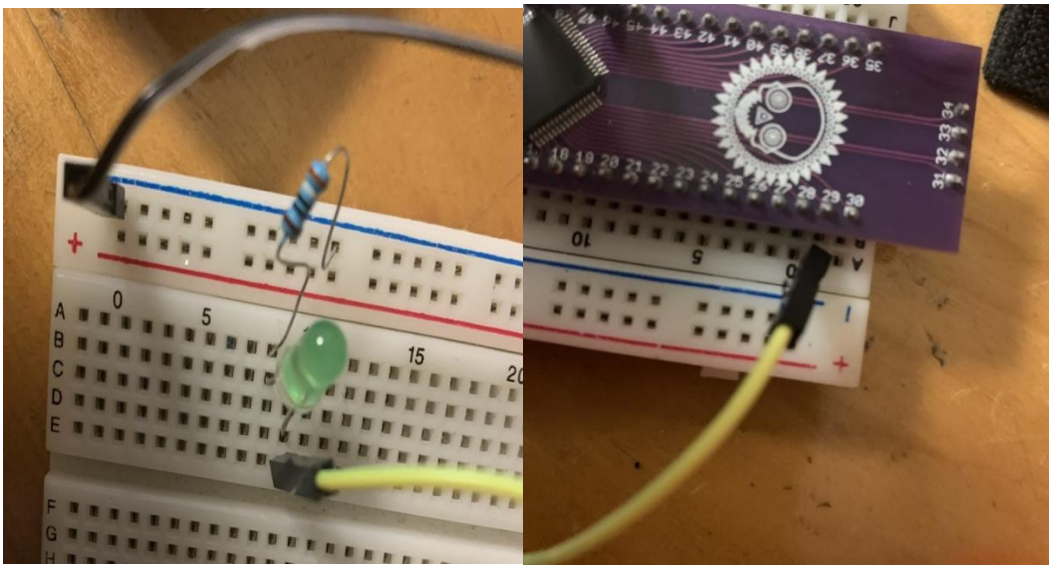
Next, the MCU should be placed along the middle of the breadboard at the top. The jumper wires connected to the 3.3V and GND must be connected to the power and ground busses on the breadboard respectively. The SWIDO wire should be connected to pin 46 and the SWCLK wire should be connected to pin 49 on the MCU.



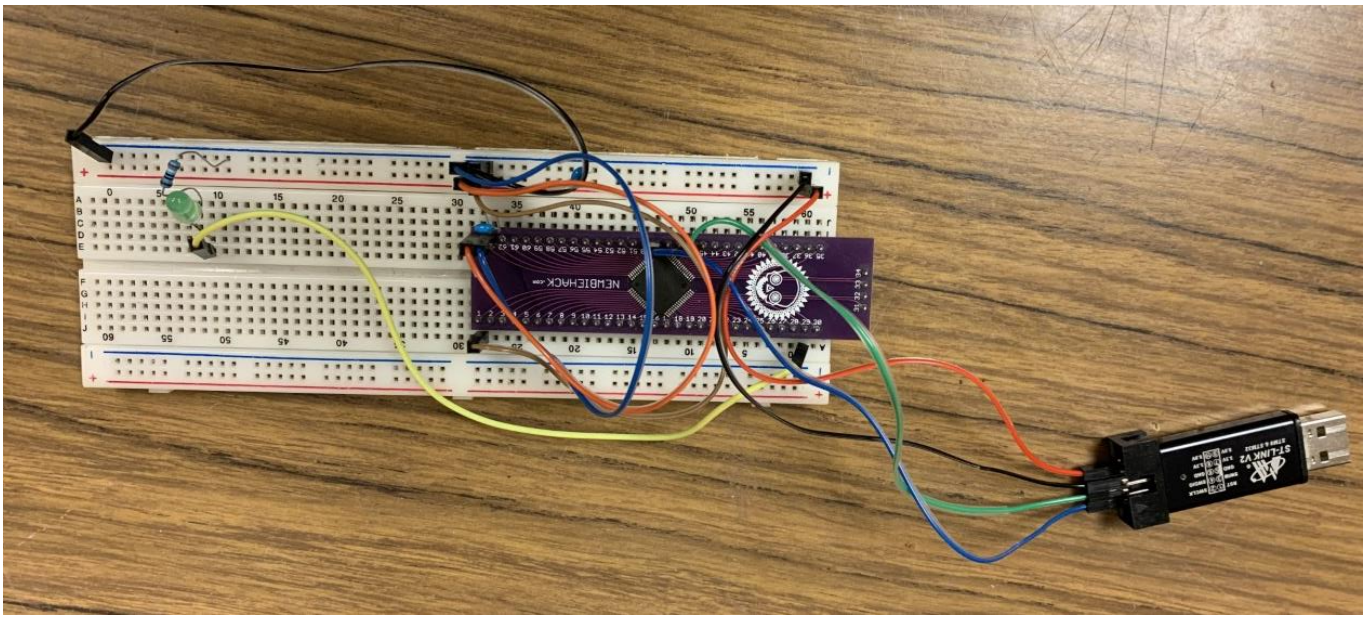
After, place 2 wires in the ground bus, 2 wires in the power bus, and a capacitor connected to both the ground and power buses. Connect the wires in the power bus to pins 64 and 1 and connect one of the wires in the ground bus to pin 63. Put legs of the last remaining capacitor in connection with pins 63 and 64.



Connect the end of the left-over wire in the ground bus to the ground bus after the break in the breadboard. Connect one leg of the resistor to the ground bus and the other on any point on the breadboard other than the ground bus or the power bus. Connect the LED in series with the resistor on the breadboard and place a wire also in series with the two. Place the end of the wire in series with the resistor and LED at pin 29 of the MCU.



The final product should look like the image below:



Uploading To the MCU

With the wiring and programing now done the code can be uploaded to the MCU. Plug the ST-Link into the computer and open the main.c in Atollic. Click “Debug” at the top of the IDE and the code should be uploaded to the MCU within a few seconds.

The program will stop at a specific section until you click “resume.”

If everything was done correctly the LED light should be blinking continuously.