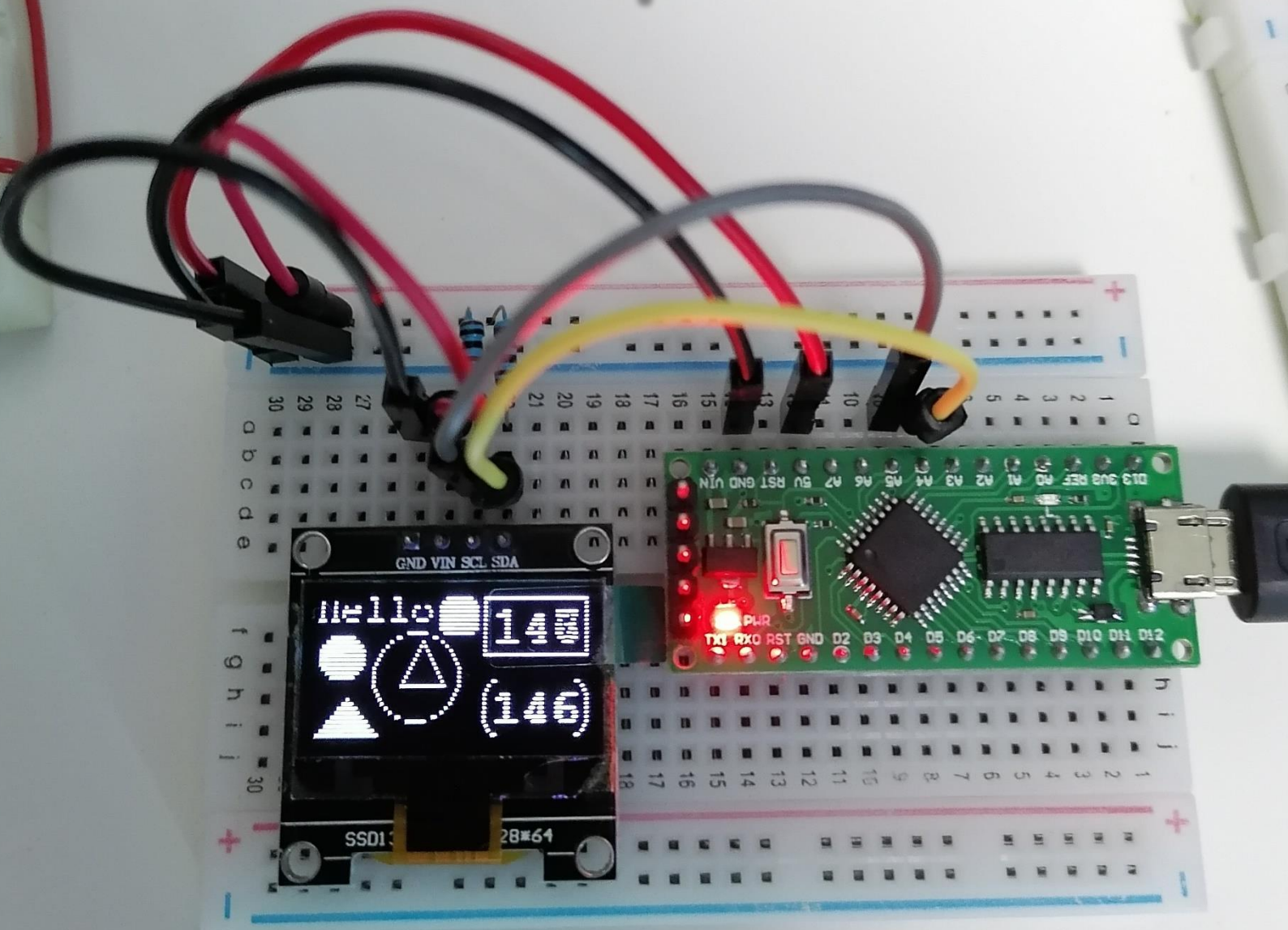




LGT8F328P

by Ricky Gai
Revision 1.6



JANUARY 18, 2021

Nexuz Innovation, Malaysia.
(MA0255412-M)

Introduction

About this Book

This is a hands-on information documented to illustrate the use of LGT8F328P chips that has been tested and run successfully on selected Arduino sketch projects using Arduino C/C++ programming language.

In the learning of Arduino and compatible clone chips, often lack of information, incomplete online comments and the circuitry implemented may not be the same as the original circuit diagram, all these problems eventually living me to the darkness.

Therefore, I decided to write this book to document issues encountered and resolved via physical trial and error approach. It is not perfect, but with necessary guidelines to get through and move on.

To understand this book, it required some basic knowledge of electronics fundamental, PCB design and C/C++ software programming.

About the Author

Ricky Gai

is the founder and technical director of Nexuz Innovation, a small R&D IT company established in Kuala Lumpur, Malaysia.

After receiving certification from Oxford Computer Engineering discipline in 1992, my career was mostly exposed to C/C++ system software development for decades about 30 years, since the MS-DOS time until today's Windows environments including real-time, networking, file system, 2D/3D games, software driver, application and mobile programming.

Nevertheless, much spare time devoted to further the electronics studies for two years before coming to Arduino platform, and my wife often staring at me. Arduino programming reminded me the MS-DOS season, it brought back memory of something like interrupt, vector and bootsector (eg. Bootloader in Arduino).

All the reference materials and source code are available via Github at:

<https://github.com/rickygai/arduino>

For any errors found, suggestions and questions, please do email to:

support@nexuzinnovation.com

Well, passion is everything and the key to success, I hope you find something useful here.

DISCLAIMER

Abbreviation	Descriptions
NEXUZ INNOVATION / AUTHOR	refers to the author, Ricky Gai.
READER / READER(S) / READER'S	refers to the person who read or knowledge transferred, accessed the circuitry setup based on the contents illustrated in this document.
COMPONENTS / EQUIPMENTS	refers to electronics components, tools, materials that used as part of the circuitry setup.
CONTENTS	Information described within the document, including software and hardware solutions or methods described by the author.
IP / INTELLECTUAL PROPERTY / COPYRIGHT / PERMISSION	refers to the copyrighted materials (eg. Photo, Diagram, Source Code, Links) that owned by other creators.

THIS CONTENTS OF THIS DOCUMENT IS SOLELY BASED ON THE ELECTRONICS COMPONENTS OR MATERIALS ADOPTED AND TESTED BY AND AT THE AUTHOR'S PREMISES. DUE TO VOLTAGE SUPPLY VARIES FROM DIFFERENT COUNTRIES AND PHYSICAL SPECIFICATION OF COMPONENTS MAY CHANGE FROM TIME TO TIME, READER(S) TAKE OWN RESPONSIBILITY TO ACCESS THE EXPERIMENTAL SOLUTIONS BASED ON THE INFORMATION DESCRIBED IN THIS DOCUMENT.

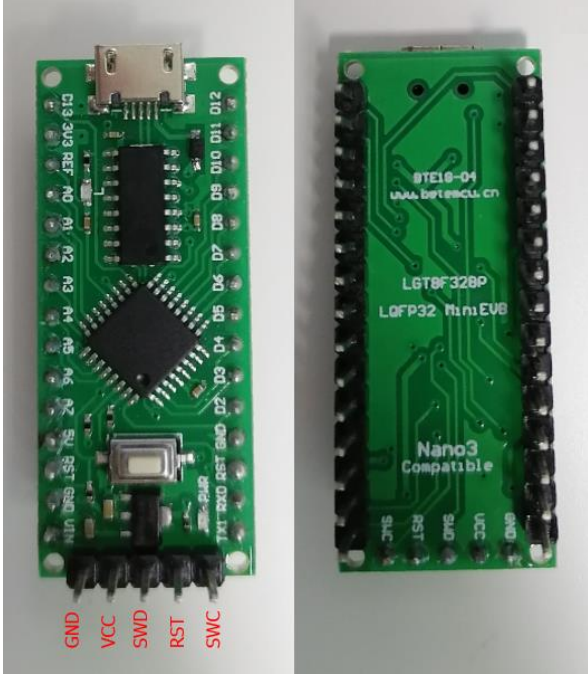
THE COMPANY NEXUZ INNOVATION AND THE AUTHOR BARES NO RESPONSIBILITY UPON ANY DAMAGE OR HARM IN CASE HAPPENS TO THE READER(S) OR READER'S SIDE, THIS INCLUDE READER'S RELATED SUCH AS HUMAN HEALTH, HARDWARE EQUIPMENTS AND OTHER LOSSES.

NEXUZ INNOVATION IS A SOLE PROPRIETORSHIP COMPANY WITH NO RELATION TO ANY HARDWARE MANUFACTURERS OR VENDORS MENTIONED IN THIS DOCUMENT, SUCH AS LOGIC GREEN, ARDUINO AND ETC. THE MENTIONED OF INTEGRATED CIRCUIT (IC) OR CHIPS PRODUCT MODELS ARE SOLELY FOR RESEARCH AND DEVELOPMENT PURPOSES ONLY.

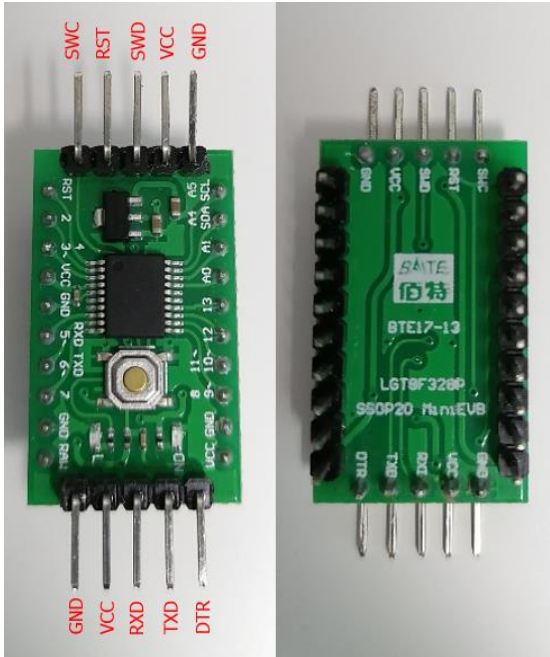
THE AUTHOR RESPECTS THE INTELLECTUAL PROPERTY FROM OTHER CREATORS, THIS DOCUMENTATION MAY SHARE SOME PARTLY EXTRACTED PORTION OF PHOTO OR DIAGRAM AS PART OF THE ILLUSTRATION USAGE. IF THERE IS ANY COPYRIGHT INFRINGEMENT, PLEASE DO INFORM THE AUTHOR TO EXCLUDE FROM THIS DOCUMENT.

Types of LGT8F328P chips

LGT8F328P LQFP32 MiniEVB

Pinout Diagram	References
	<p>LGT8F328P is one of the Arduino clone chip created by Logic Green.</p> <p>Logic Green LGT8F328P datasheet (in Chinese)</p> <p>Logic Green official manuals (datasheets, circuit pinout diagrams)</p> <p>Ralph Bacon LGT8F328P-Arduino-Clone-Chip-ATMega328P (with English version of LGT8F328P datasheet)</p>

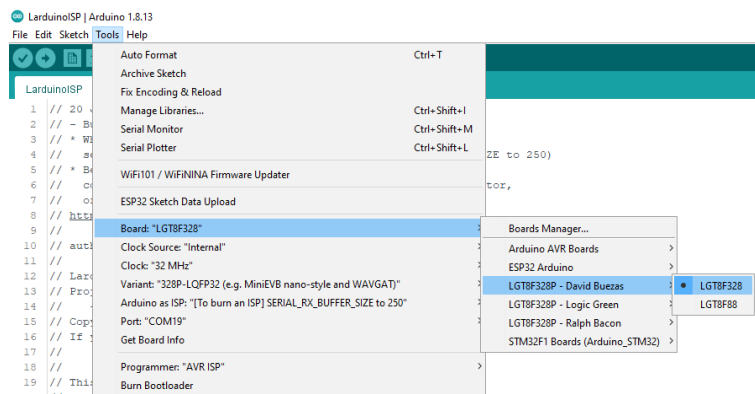
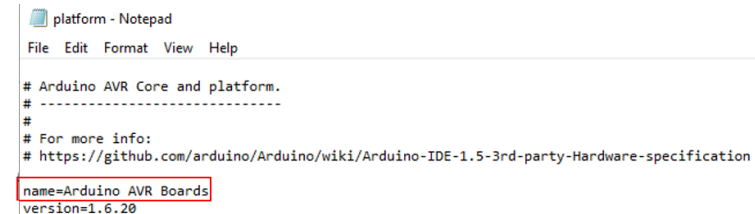
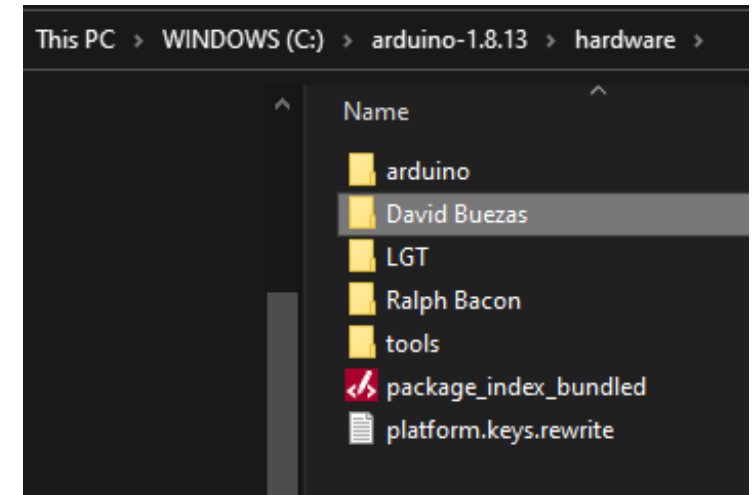
LGT8F328P SSOP20 MiniEVB

Pinout Diagram	References
	<p>Same as above.</p>

Prerequisite

Arduino IDE - Setup the LGT8F328P library

Download, install and configure the LGT8F328P libraries

Diagrams	Descriptions & References
<p>It is very important to configure the installed LGT8F328P libraries as to appear under the Arduino IDE menu outstandingly such as below:</p>  <p>Figure 1: Multiple LGT8F328P libraries installed with organized</p>  <p>Figure 2: The default attribute name="Arduino AVR Boards"</p>  <p>Figure 3: The default attribute name="Arduino AVR Boards"</p>	<p>There are many versions of LGT8F328P libraries built by different authors. Here, I choose to install three board libraries from:</p> <p>Logic Green Ralph Bacon David Buezas</p> <p>To have a proper names listed as shown on Figure 1, you have to edit the file platform.txt from each of the author's library path installed, otherwise the listed items under "Board Manager" will be confusing such as below:</p> <p>Arduino AVR Boards Arduino AVR Boards Logic Green Arduino AVR Compatible Boards</p> <p>Depends of which version of Arduino IDE you are using, mine is 1.8.13 and it duplicated the default name "Arduino AVR Boards" because the platform.txt bundled under Logic Green and Ralph Bacon package file used the same default attribute "name=Arduino AVR Boards" as shown on Figure 2.</p> <p>To solve this problem, I edited three of the platform.txt files and renamed the attribute "name=" of each file respectively to something like below:</p> <p>name=LGT8F328P - Logic Green name=LGT8F328P - Ralph Bacon name=LGT8F328P - David Buezas</p> <p>To be more precisely, I renamed the library folder for Ralph Bacon and David Buezas as shown on Figure 3.</p> <p>In my case, these three platform.txt files are located under:</p> <p>C:\arduino-1.8.13\hardware\LGT\avr\platform.txt C:\arduino-1.8.13\hardware\Ralph Bacon\avr\platform.txt C:\arduino-1.8.13\hardware\David Buezas\avr\platform.txt</p> <p>The path locations above may be different, subjected to individuals' Arduino IDE installation.</p>

Arduino Sketch Projects

[LarduinoISP - How to burn the BOOTLOADER ?](#)

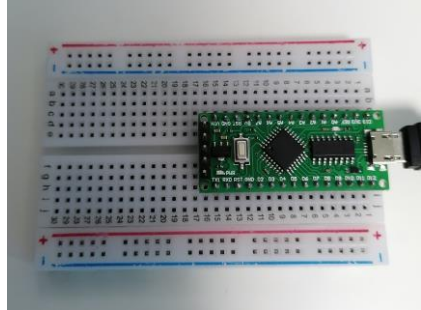
Introduction

Diagrams	Descriptions & References
<p>Figure 4: LarduinoISP working circuitry setup</p>	<p>LarduinoISP is the use of Arduino ATmega328P (eg. NANO, UNO) or LGT8F328P (LQFP32 MiniEVB) as ISP Programmer to burn BOOTLOADER to the Target LGT8F328P SSOP20 MiniEVB as shown at Figure 4.</p> <p>Here, I want to share some painful experiences that went through in dealing with the LarduinoISP circuitry setup.</p> <p>I started by referring to LGTMCU/LarduinoISP, at that time I was doubted about the short-circuit of the 10kΩ resistor between VCC and SWD pin as shown at Figure 5, with two green highlighted circles below:</p> <p>Figure 5: LGT/LarduinoISP extracted diagram</p> <p>Really wasted much time on it, no one from the Arduino Forum can gives me at least a snapshot photo of how to setup the physical circuitry correctly.</p> <p>Based on Figure 4, I decided to remove the SWD (blue wire) from connecting to the 10kΩ pull-up resistor and connect directly to pin 12 of the ISP Programmer.</p> <p>Next, I used David Buezas's LarduinoISP sketch to upload to the ISP Programmer because his sketch is more recently updated.</p> <p>Somehow, still failed to upload the sketch (may be that time Arduino Nano was used as ISP Programmer). Later, a site called SuperUserNameMan/LGTISP, after reading its usage section, I begin to realize the whole story and quickly edit the HardwareSerial.h to to change the instruction from:</p> <pre>#define SERIAL_RX_BUFFER_SIZE 64</pre> <p>to:</p> <pre>#define SERIAL_RX_BUFFER_SIZE 250</pre> <p>Eventually, LarduinoISP successfully burned the BOOTLOADER to the Target LGT8F328P SSOP20 MiniEVB.</p> <p>The entire circuitry is very straight forward actually but took me sometime to accomplish.</p> <p>We will cover more details about the hands-on process on next page.</p>

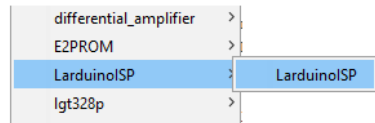

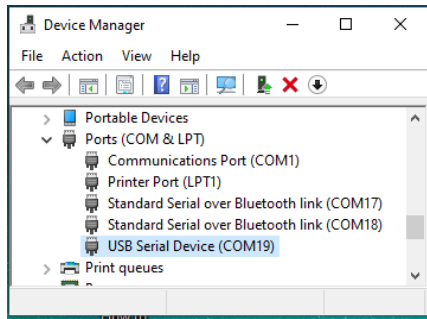
The Parts list

Components	Quantity
LGT8F328P LQFP32 MiniEVb	1
LGT8F328P SSOP20 MiniEVb	1
FTDI FT232RL	1
RESISTOR 10kΩ	1
10 cm Female-to-Male Jumper Wire	5
10 cm Female-to-Female Jumper Wire	5
10 cm Male-to-Male Jumper Wire	1
USB 2.0 Micro B cable connector	1
USB 2.0 Mini B cable connector	1
MB102 Mini Breadboard 8.5CM x 5.5CM 400 Holes	2

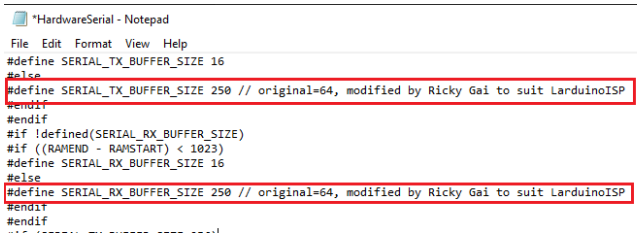
Installing the LGT8F328P board library

Illustrations	Descriptions
<ol style="list-style-type: none"> First, you have to complete the Download, install and configure the LGT8F328P libraries stage. For illustration purposes, we adopted David Buezas LGT8FX board library. After you have completed step 1, put the LGT8F328P LQFP32 MiniEVb on the breadboard then connect the USB Micro B cable as shown on Figure 6 to the PC. 	 <p>Figure 6: LGT8F328P LQFP32 MiniEVb on empty breadboard</p>

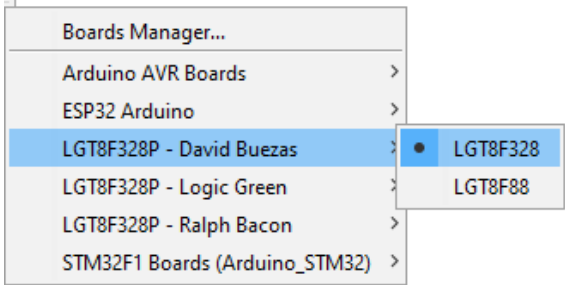

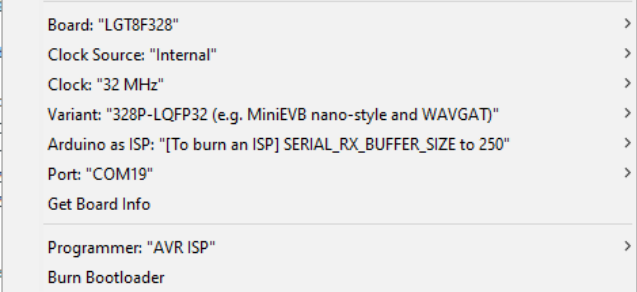
Opening the LarduinoISP sketch

Illustrations	Descriptions
<ol style="list-style-type: none"> At the Arduino IDE, File → Examples → LarduinoISP → LarduinoISP, click on it to open such as Figure 7. 	 <p>Figure 7: Opening the LarduinoISP sketch</p>
<ol style="list-style-type: none"> Next, turn ON the LGT8F328P LQFP32 MiniEVb, the respective serial port should be detected, in this case it is COM19 under the Windows Device Manager at Figure 9. Goto the File menu, Tools → Port: → select COM19 as shown below:  <p>Figure 8: Selecting the LGT8F328P LQFP32 Serial Port</p>	 <p>Figure 9: LGT8F328P LQFP32 Serial Port under Device Manager</p>

Updating HardwareSerial.H

Illustrations	Descriptions and Fundamental Units
<p>6. Open the file HardwareSerial.H using Notepad:</p> <p>C:/arduino-1.8.13/hardware/arduino/avr/cores/arduino/HardwareSerial.h</p> <p>7. Change "#define SERIAL_RX_BUFFER_SIZE 64" to "#define SERIAL_RX_BUFFER_SIZE 250" as shown on Figure 10.</p> <p>8. Save the file.</p>	 <p>The screenshot shows the HardwareSerial.H file in Notepad. Two lines are highlighted with red boxes: "#define SERIAL_TX_BUFFER_SIZE 250 // original=64, modified by Ricky Gai to suit LarduinISP" and "#define SERIAL_RX_BUFFER_SIZE 250 // original=64, modified by Ricky Gai to suit LarduinISP".</p> <p>Figure 10: HardwareSerial.H updates</p>

Uploading LarduinISP sketch to ISP Programmer (LGT8F328P LGFP32 MiniEVB)

Illustrations	Descriptions and Fundamental Units
<p>9. Goto File menu, Tools → Board: → LGT8F328P - David Buezas → select LGT8F328 as shown on Figure 11.</p>	 <p>The screenshot shows the Boards Manager window. Under the 'Arduino AVR Boards' category, 'LGT8F328P - David Buezas' is selected. A sub-menu is open showing 'LGT8F328' and 'LGT8F88', with 'LGT8F328' being the selected option.</p> <p>Figure 11: Selecting David Buezas LGT8F328 board library</p>
<p>10. Make sure under the Tools menu options, all settings are based on Figure 12.</p> <p>11. Upload the sketch by clicking  button.</p>	 <p>The screenshot shows the Tools menu settings for the LGT8F328 board. The settings are: Board: "LGT8F328", Clock Source: "Internal", Clock: "32 MHz", Variant: "328P-LQFP32 (e.g. MiniEVB nano-style and WAVGAT)", Arduino as ISP: "[To burn an ISP] SERIAL_RX_BUFFER_SIZE to 250", Port: "COM19", Get Board Info, Programmer: "AVR ISP", and Burn Bootloader.</p> <p>Figure 12: David Buezas LGT8F328 settings</p>

The success messages of LarduinoISP sketch uploaded to ISP Programmer.

Descriptions
<p>12. If the LarduinoISP sketch is successfully uploaded to ISP Programmer (LGT8F328P LQFP32 MiniEVB), you should be seeing messages below:</p> <pre> avrdude: AVR device initialized and ready to accept instructions Reading ##### 100% 0.00s avrdude: Device signature = 0x1e950f (probably m328p) avrdude: reading input file "C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_659691\LarduinoISP.ino.hex" avrdude: writing flash (5420 bytes): Writing ##### 100% 1.34s avrdude: 5420 bytes of flash written avrdude: verifying flash memory against C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_659691\LarduinoISP.ino.hex: avrdude: load data flash data from input file C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_659691\LarduinoISP.ino.hex: avrdude: input file C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_659691\LarduinoISP.ino.hex contains 5420 bytes avrdude: reading on-chip flash data: Reading ##### 100% 1.05s avrdude: verifying ... avrdude: 5420 bytes of flash verified avrdude done. Thank you. </pre>

Figure 13: The success messages of LarduinoISP sketch uploaded to ISP Programmer

Wiring up the LarduinoISP circuitry

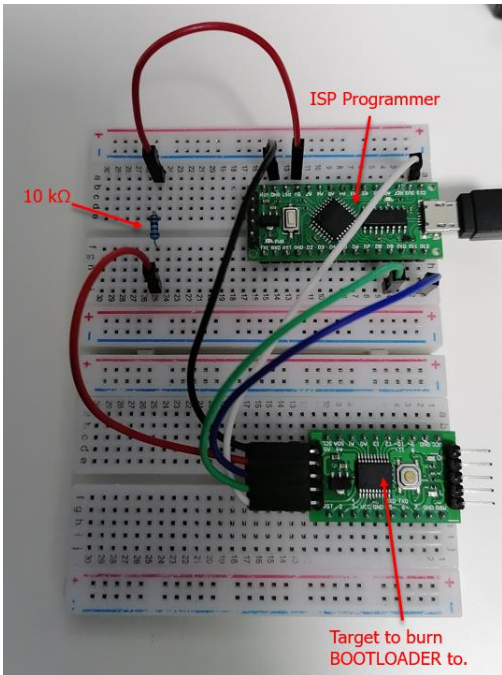
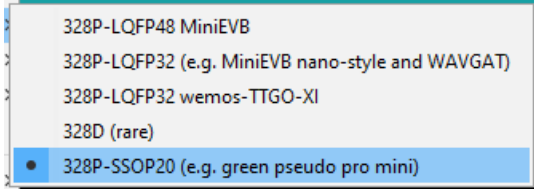
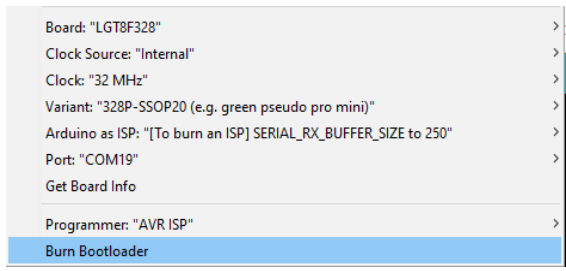
Illustrations	Descriptions												
<p>13. Now, the ISP Programmer is ready.</p> <p>14. Turn off the ISP Programmer.</p> <p>15. We are going to wire up the LarduinoISP circuitry just like Figure 15.</p> <p>16. LarduinoISP wire connection table:</p> <table border="1"> <thead> <tr> <th>LGT8F32P SSOP20 (Target)</th> <th>LGT8F328 LQFP32 (ISP Programmer)</th> </tr> </thead> <tbody> <tr> <td>GND</td> <td>GND</td> </tr> <tr> <td>VCC</td> <td>5V</td> </tr> <tr> <td>SWD</td> <td>D12</td> </tr> <tr> <td>RST</td> <td>D10</td> </tr> <tr> <td>SWC</td> <td>D13</td> </tr> </tbody> </table>	LGT8F32P SSOP20 (Target)	LGT8F328 LQFP32 (ISP Programmer)	GND	GND	VCC	5V	SWD	D12	RST	D10	SWC	D13	 <p>Figure 15: LarduinoISP circuitry setup</p>
LGT8F32P SSOP20 (Target)	LGT8F328 LQFP32 (ISP Programmer)												
GND	GND												
VCC	5V												
SWD	D12												
RST	D10												
SWC	D13												

Figure 14: LarduinoISP wire connection

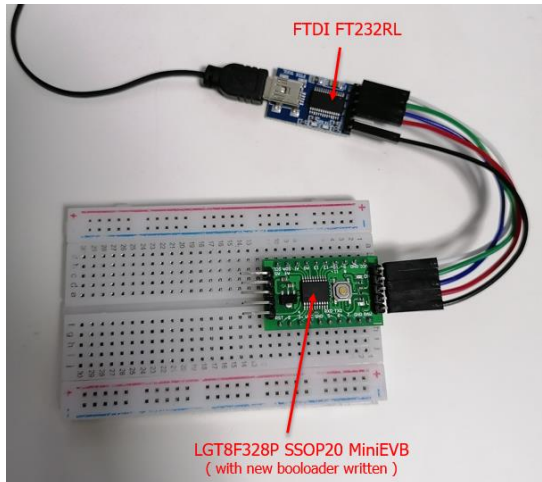
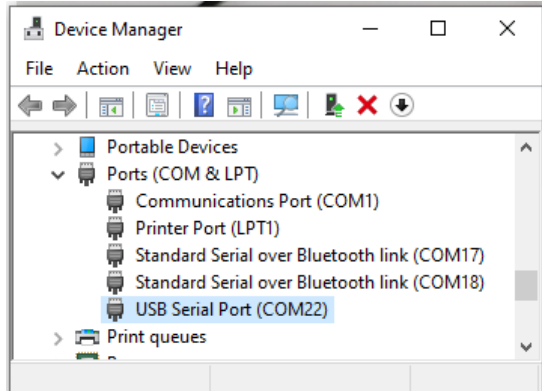
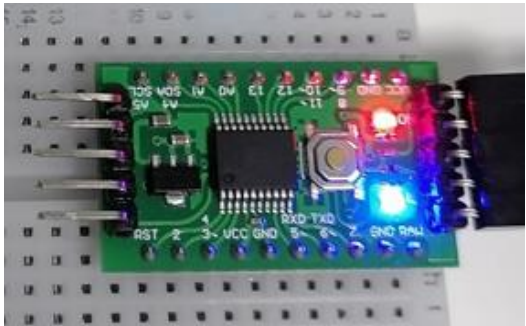
To burn BOOTLOADER to Target LGT8F328P SSOP20 MiniEVb

Illustrations	Descriptions
<p>17. We are now going to burn the BOOTLOADER to the Target LGT8F328P SSOP20 MiniEVb.</p> <p>18. Turn on the ISP Programmer.</p> <p>19. At the File menu, Tools → Variant: → select "328P-SSOP20 (e.g. green pseudo pro mini)" as shown on Figure 16.</p> <p>20. Ignore the "Arduino as ISP", select Programmer as "AVR ISP".</p> <p>21. Click on the "Burn Bootloader".</p>	 <p>Figure 16: Change the Variant to Target LGT8F328P SSOP20</p>  <p>Figure 17: To Burn Bootloader</p>


The success messages of burning the BOOTLOADER to the Target.

Descriptions
<p>22. If the BOOTLOADER is successful, you will see the message below:</p> <pre> avrdude: AVR device initialized and ready to accept instructions Reading ##### 100% 0.00s avrdude: Device signature = 0x1e950f (probably m328p) avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed To disable this feature, specify the -D option. avrdude: erasing chip avrdude: reading input file "C:\arduino-1.8.13\hardware\David Buezas\avr\bootloaders\lgt8fx8ps20\optiboot_lgt8f328ps20.hex" avrdude: writing flash (30720 bytes): Writing ##### 100% 0.25s avrdude: 30720 bytes of flash written avrdude: verifying flash memory against C:\arduino-1.8.13\hardware\David Buezas\avr\bootloaders\lgt8fx8ps20\optiboot_lgt8f328ps20.hex: avrdude: load data flash data from input file C:\arduino-1.8.13\hardware\David Buezas\avr\bootloaders\lgt8fx8ps20\optiboot_lgt8f328ps20.hex: avrdude: input file C:\arduino-1.8.13\hardware\David Buezas\avr\bootloaders\lgt8fx8ps20\optiboot_lgt8f328ps20.hex contains 30720 bytes avrdude: reading on-chip flash data: Reading ##### 100% 0.02s avrdude: verifying ... avrdude: 30720 bytes of flash verified avrdude: reading input file "0x3f" avrdude: writing lock (1 bytes): Writing ##### 100% 0.01s avrdude: 1 bytes of lock written avrdude: verifying lock memory against 0x3f: avrdude: load data lock data from input file 0x3f: avrdude: input file 0x3f contains 1 bytes avrdude: reading on-chip lock data: Reading ##### 100% 0.00s avrdude: verifying ... avrdude: 1 bytes of lock verified avrdude done. Thank you. </pre> <p>Figure 18: The success messages of burning BOOTLOADER to Target LGT8F328P SSOP20 MiniEVb</p>

Testing the LGT8F328P SSOP20 MiniEVB with new BOOTLOADER burned.

Illustrations	Descriptions and Fundamental Units												
<p>23. Congratulations, the Target LGT8F328P SSOP20 MiniEVB is burned with new BOOTLOADER image.</p> <p>24. Here, we are going to do a test on it by uploading a fast blinking sketch to see the significant changing effects. If it is uploaded successfully, the LGT8F328P SSOP20 MiniEVB should be blinking faster at 64 ms interval.</p> <p>25. Turn off the ISP Programmer, connect the LGT8F328P SSOP20 MiniEVB to FTDI FT232RL as shown on Figure 20.</p> <p>26. The FTDI FT232RL connection, its CTS pin is not required here:</p> <table border="1"> <thead> <tr> <th>LGT8F32P SSOP20</th><th>FTDI FT232RL</th></tr> </thead> <tbody> <tr> <td>GND</td><td>GND</td></tr> <tr> <td>VCC</td><td>5V</td></tr> <tr> <td>RXD</td><td>TXD</td></tr> <tr> <td>TXD</td><td>RXD</td></tr> <tr> <td>DTR</td><td>DTR</td></tr> </tbody> </table> <p>Figure 19: The standalone LGT8F328P SSOP20 MiniEVB pin connection with FTDI FT232RL</p> <p>27. Turn on the FTDI FT232RL that will power up LGT8F328P SSOP20 MiniEVB.</p> <p>28. NOTE: This will create a new serial port COM22 interface as shown on Figure 21.</p>	LGT8F32P SSOP20	FTDI FT232RL	GND	GND	VCC	5V	RXD	TXD	TXD	RXD	DTR	DTR	 <p>Figure 20: The standalone LGT8F328P SSOP20 MiniEVB with FTDI FT232RL interface</p>  <p>Figure 21: FTDI FT232RL new serial port interface</p>  <p>Figure 22: FTDI FT232RL default blinking (blue light) two seconds interval</p>
LGT8F32P SSOP20	FTDI FT232RL												
GND	GND												
VCC	5V												
RXD	TXD												
TXD	RXD												
DTR	DTR												
<p>29. Take a look at the LGT8F328P SSOP20 MiniEVB by default itself, it is blinking at internal roughly two seconds as shown on Figure 22.</p> <p>To see the different, later when the new sketch blink64ms.ino is uploaded to it, the blue LED should be blinking faster at 64 ms interval.</p>													

Uploading blink64ms.ino sketch to LGT8F328P SSOP20 MiniEVb via FT232RL

Descriptions
<p>30. Here, we are to upload a sketch called blink64ms.ino to the LGT8F328P SSOP20 MiniEVb since the Bootloader already written, if it is working normally after a reset, the Bootloader should starts and jumps to the program address where the blink64ms.ino program will be loaded and run.</p> <p>31. Select the correct Port COM22 (FT232RL) from the menu as shown below:</p> <div data-bbox="443 468 1133 788" data-label="Image"> <p>The screenshot shows the 'Tools' menu in the Arduino IDE. The 'Board' is set to 'LGT8F328P', 'Clock Source' is 'Internal', 'Clock' is '32 MHz', 'Variant' is '328P-SSOP20 (e.g. green pseudo pro mini)', 'Arduino as ISP' is '[To burn an ISP] SERIAL_RX_BUFFER_SIZE to 250', 'Port' is 'COM22', 'Get Board Info' is selected, 'Programmer' is 'AVR ISP', and 'Burn Bootloader' is at the bottom.</p> </div> <p>Figure 23: Selecting the FTDI FT232RL serial port COM22</p> <p>32. At Arduino IDE, open the blink64ms.ino sketch then click the  button to upload sketch.</p> <p>33. If it is uploaded successfully you will notice the blue LED will blink faster continuously.</p>

The success messages of uploading blink64ms.ino to LGT8F328P SSOP20 MiniEVb

Descriptions and Fundamental Units
<p>34. The success messages after blink64ms.ino sketch uploaded to LGT8F328P SSOP20 MiniEVb will look like below:</p> <div data-bbox="202 1200 1418 1704" data-label="Image"> <p>The screenshot shows the terminal output of the Arduino IDE. It displays the following messages: 'avrdude: AVR device initialized and ready to accept instructions', 'Reading ##### 100% 0.00s', 'avrdude: Device signature = 0x1e950f (probably m328p)', 'avrdude: reading input file "C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_82099\blink64ms.ino.hex"', 'avrdude: writing flash (1104 bytes):', 'Writing ##### 100% 0.48s', 'avrdude: 1104 bytes of flash written', 'avrdude: verifying flash memory against C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_82099\blink64ms.ino.hex:', 'avrdude: load data flash data from input file C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_82099\blink64ms.ino.hex:', 'avrdude: input file C:\Users\RICKYG~1\AppData\Local\Temp\arduino_build_82099\blink64ms.ino.hex contains 1104 bytes', 'avrdude: reading on-chip flash data:', 'Reading ##### 100% 0.47s', 'avrdude: verifying ...', 'avrdude: 1104 bytes of flash verified', and 'avrdude done. Thank you.'</p> </div> <p>Figure 24: The success messages of uploading blink64ms.ino sketch to LGT8F328P SSOP20 MiniEVb</p> <p>35. DONE.</p>