IoT-ALE: Demystifying MCUs with Arduino

Nova (aka Justin) King

SCaLE 17x - March 2019





FIGURE 1. The Sunbeam Radiant Control toaster.



Arduino

• ATMEGA328

• Key parameters [edit]

Parameter	Value		
CPU type	8-bit AVR		
Performance	20 MIPS at 20 MHz ^[2]		
Flash memory	32 kB		
SRAM	2 kB		
EEPROM	1 kB		
Pin count	28 or 32 pin: PDIP-28, MLF-28, TQFP-32, MLF-32 ^[2]		
Maximum operating frequency	20 MHz		
Number of touch channels	16		
Hardware QTouch Acquisition	No		
Maximum I/O pins	23		
External interrupts	2		
USB Interface	No		
USB Speed	-		

ESP8266

- Processor: L106 32-bit <u>RISC</u> microprocessor core based on the <u>Tensilica</u> Xtensa Diamond Standard 106Micro running at 80 MHz^[5]
- Memory:
 - 32 KiB instruction RAM
 - 32 KiB instruction cache RAM
 - 80 KiB user-data RAM
 - 16 KiB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- <u>IEEE 802.11</u> b/g/n <u>Wi-Fi</u>
 - Integrated <u>TR switch</u>, <u>balun</u>, <u>LNA</u>, <u>power amplifier</u> and <u>matching network</u>
 - <u>WEP</u> or <u>WPA/WPA2</u> authentication, or open networks
- 16 <u>GPIO</u> pins
- <u>SPI</u>
- <u>I²C</u> (software implementation)^[6]
- <u>I²S</u> interfaces with DMA (sharing pins with GPIO)
- <u>UART</u> on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit <u>ADC</u> (successive approximation ADC)

ESP32

- Processors:
 - CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 <u>DMIPS</u>
 - Ultra low power (ULP) co-processor
- Memory: 520 KiB SRAM
- Wireless connectivity:
 - Wi-Fi: <u>802.11</u> b/g/n
 - Bluetooth: v4.2 BR/EDR and BLE
- Peripheral interfaces:
 - 12-bit <u>SAR ADC</u> up to 18 channels
 - 2 × 8-bit <u>DACs</u>
 - 10 × touch sensors (<u>capacitive sensing</u> GPIOs)
 - Temperature sensor
 - 4 × <u>SPI</u>
 - $2 \times \underline{I^2S}$ interfaces
 - $2 \times \underline{I^2C}$ interfaces
 - 3 × <u>UART</u>
 - •

ESP32 Con't

- <u>SD/SDIO/CE-ATA/MMC/eMMC</u> host controller
- SDIO/SPI slave controller
- <u>Ethernet MAC interface with dedicated DMA and IEEE 1588 Precision Time</u>
 <u>Protocol</u> support
- <u>CAN bus</u> 2.0
- Infrared remote controller (TX/RX, up to 8 channels)
- Motor <u>PWM</u>
- LED <u>PWM</u> (up to 16 channels)
- Hall effect sensor
- Ultra low power analog pre-amplifier
 - •

ESP32 Con't

- security:
 - IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
 - Secure boot
 - Flash encryption
 - 1024-bit OTP, up to 768-bit for customers
 - Cryptographic hardware acceleration: <u>AES</u>, <u>SHA-2</u>, <u>RSA</u>, <u>elliptic</u> <u>curve cryptography</u> (ECC), <u>random number generator</u> (RNG)
- Power management:
 - Internal <u>low-dropout regulator</u>
 - Individual power domain for RTC
 - 5uA deep sleep current
 - Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

Your devices and networking



Hybrid solution: Local access + cloud access

Your devices and networking



Cloud only access, no local network

Your devices and networking

No access because you forgot to install the wifi firmware :)



Labs

- IDE/Board Setup
 - Install Python if needed
 - Install Arduino IDE
 - Install ESP32 board interface
- Blinky
 - \circ $\,$ $\,$ Open and upload to board $\,$
- WiFi
 - Open from examples menu
 - Upload
- Sensors
 - Install library from library manager
 - Open example
 - Modify example to work with the current board

IDE Setup

https://www.arduino.cc/en/Guide/Linux

- sudo chmod 666 /dev/ttyUSB0 if it won't upload
- <u>https://www.arduino.cc/en/Guide/Windows</u>

• <u>https://www.arduino.cc/en/Guide/MacOSX</u>

Setting up the ESP32 board drivers

(https://dl.espressif.com/dl/package_esp32_index.json)

▼	Preferences	×
Settings Network		
Skatchback location:		
/home/kabsox/Arduino/Sketch		Browse
Editor language:	System Default v (requires restart of Arduino)	
Editor font size:	12	
Interface scale:	Automatic 100 💭 % (requires restart of Arduino)	
Theme:	Default theme 💌 (requires restart of Arduino)	
Show verbose output during:	Compilation upload	
Compiler warnings:	None 🔻	
Display line numbers		
Enable Code Folding		
Verify code after upload		
Use external editor		
Aggressively cache compile	ed core	
Check for updates on start	dr	
✓ Update sketch files to new	extension on save (.pde -> .ino)	
Save when verifying or uplo	pading	
Additional Boards Manager UR	Ls: https://dl.espressif.com/dl/package_esp32_index.json	
More preferences can be edite	d directly in the file	
/home/ka6sox/.arduino15/pref	erences.txt	
(edit only when Arduino is not r	unning)	
	OK	Cancel

Blinky

- File > Open
- Open the Blinky file in the Blinky folder
- Upload the program to the board
- sudo chmod 666 /dev/ttyUSB0 if it won't upload







Wifi Scan

New Open Open Recent	Ctrl+N Ctrl+O	•			
Examples					
Close Save Save As Page Setup	Ctrl+W Ctrl+S Ctrl+Shift+S Ctrl+Shift+P		Examples for Node32s ArduinoOTA BluetoothSerial DNSServer EEPROM	* * *	3
Print Preferences	Ctrl+P Ctrl+Comma		ESP32 ESP32 Async UDP		ETH_LAN8720 ETH_LAN8720 internal_clock
Quit	Ctrl+Q		ESP32 Azure IoT Arduino	1	ETH_TLK110
			ESPS2 BLE Arduno ESPmDNS		WiFiBlueToothSwitch
			HTTPClient		WiFiClient
			NetBIOS	1	WiFiClientBasic
			Preferences	1	WiFiClientEnterprise
			SD(esp32)	•	WiFiClientEvents
			SD_MMC	1	WiFiClientStaticIP
			SimpleBLE	1	WiFiIPv6
			SPI	1	WiFiMulti
			SPIFFS	1	WiFiScan
			Ticker	1	WiFiSmartConfig
			Update	1	WiFiTeInetToSerial
		-	WebServer	1	WiFiUDPClient
			WiFi	1	WPS

Wifi Monitor (Tools > Serial Monitor)

💿 СОМ10	
	Send
scan start	
scan done	
2 networks found	
1: lfevents (-78)*	
	Change to 115200 baud
V Autoscroll Show timestamp	Newline 👻 115200 baud 👻 Clear output

Library Manager (for sensors)

🥺 WiFiScan∣Arduin File Edit Sketch To	o 1.8.7 pols Help		
WiFiScan	Auto Format Archive Sketch Fix Encoding & Reload	Ctrl+T	₽
/*	Manage Libraries	Ctrl+Shift+I	
* This sket * The API i * the most	Serial Monitor Serial Plotter	Ctrl+Shift+M Ctrl+Shift+L	ks. ld library, file you need
*/ #include "WiF	WiFi101 Firmware Updater		E
<pre>void setup() { Serial.be // Set Wi</pre>	Board: "Node32s" Upload Speed: "921600" Flash Frequency: "80MHz" Port: "COM10" Get Board Info)))	an AP if it we
WiFi.disc delay(100	Programmer: "AVRISP mkII" Burn Bootloader	•	
Serial.print	tln("Setup done");		
<	m		
Leaving Hard resetting v	ia RTS pin		Â
•	m		•
10		Node32s, 80MHz	, 921600 on COM10

Sensors Library Installation

💿 Library Manager	23
Type All 🔹 Topic All 🔹 tsl2591	
Select version Instal	*
BlueDot BME280 TSL2591 by BlueDot Version 1.0.4 INSTALLED BlueDot library for BME280 and TSL2591 sensors. Read temperature, relative humidity, pressure and illuminance with BME280 and TSL2591 sensors. More info Select version Install	>
	-
	Close

Sensors Library Example

0	WiFiScan Ardu	ino 1.8.7		
<u>F</u> ile	<u>Edit</u> <u>Sketch</u>	Tools <u>H</u> elp		
	New	Ctrl+N		₽ D
	Open	Ctrl+0		
	Open Recent	•		M.,
	Sketchbook	ſ	A	<u>^</u>
	Examples	1	ArduinoOTA	0.2 70 1
	Close	Ctrl+W	BluetoothSerial	ou need
	Save	Ctrl+S	DNSServer 🕨	=
	Save As	Ctrl+Shift+S	EEPROM •	
		01 010 B	ESP32	
	Page Setup	Ctri+Shift+P	ESP32 Async UDP	
	Print	Ctrl+P	ESP32 Azure IoT Arduino	
	Preferences	Ctrl+Comma	ESP32 BLE Arduino	F 14 11
	Ouit	Ctrl. O	ESPmDNS •	LT TC WC
	WIII . GIDCO	Cui+Q	HTTPClient •	
	delay(100)	;	NetBIOS •	
	Serial pri	ntln("Setup	Preferences •	
}	borraripri	ino in (be bup	SD(esp32)	
			SD_MMC	-
*	id loop()	, III.	SimpleBLE •	•
			SPI	
			SPIFFS •	
Hai	nd resetting	via RTS pin.	Ticker 🕨	^
			Update 🕨	+
•		m	WebServer	•
10			WiFi	n COM10
			WiFiClientSecure	
			Examples from Custom Libraries	
			BlueDot BME280 TSL2591	BME280_TSL2591_Test
			∇	

Modifying it to Work



Expected Output

	Send
20.97	*
39.11	
1019.05	
-50.26	
2248.00	
214.05	
20.97	
39.11	
1019.05	
-50.26	
2251.00	
	Newline
	20.97 39.11 1019.05 -50.26 2248.00 214.05 20.97 39.11 1019.05 -50.26 2251.00

Examples I Used:

github.com/chromenova/ sensornodeexamples

IoT-ALE: Discovering Tiny Snakes

IoT development without the need to compile (mostly)

John 'Warthog9' Hawley

SCaLE 17x - March 2019

Quick: MicroPython vs. CircuitPython?



MicroPython



Why is this different?

<pre>artialUpdateExample : example for Waveshare 1.54*, 2.31* and 2.9* e-Paper and the same e-papers from Dalian Good Display Inc. Created by Jean-Marc Zingg based on demo code from Good Display for GDEP0150C1. I The e-paper displays are available from: https://www.aliexpress.com/store/product/Mblesale-1-54inch-E-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 http://www.aliexpress.com/store/product/AproductSpath=2007_803Sproduct_id=35120 or https://www.aliexpress.com/store/product/AproductSpath=2007_803Sproduct_id=35120 or https://www.aliexpress.com/store/product/Orout=productSpath=2007_803Sproduct_id=35120 or https://www.aliexpress.com/store/product/Dromu.scr/index.php?topic=487007.0 Good Dispay ePaper for Arduino : https://forum.arduino.cc/index.php?topic=487007.0 Good Dispay ePaper for Arduino : https://forum.arduino.cc/index.php?topic=487007.0 Good Dispay ePaper for Arduino : D. S. G. > DB, CLK > DS, DH >> DF, OHD >> OHD >> 3.3V >> 3.3V mapping suggestion from Waveshare 2.9inch e-Paper to generic ESP2066 BUSY >> DE, ST >> OH, DC >> DS, CS >> DB, CLK >> DS : DH >> DF OHD >> OHD >> GND >.3.V >> 3.3V mapping suggestion for SF32, e.g. LULIN2, see/variants//pins arduino.h for your board NDTE: there are variants with different pins for SPI ! CHECK SPI PINS OF YOUR BOARD BUSY >> DC >> R. (LK >> DS, DL)> LK >> DS, DL >> AS DC >> GND >.3.V >> 3.3V mapping suggestion for AVR, UNO, NANO etc. BUSY > 7, RST >> 0, DC >> 8, CS >> 10, CLK >> DS II >> DY include <codedpdisoci 1.54*="" b="" codepdisoci.cppp="" w<br="">include <codedpdisoci 2.3*="" b="" codepdisoci.cppp="" w<br="">include <codedpdisoci 2.3*="" b="" codepdisoci.cppp="" w<br="">include <codedpdisoci 2.5*="" b="" codepdisoci.cpp="" w<br="">include <codedpdisoci 2.5*="" b="" codep<="" codepdisoci="" codepdisoci.cpp="" th="" w<=""><th></th><th></th><th>0</th></codedpdisoci></codedpdisoci></codedpdisoci></codedpdisoci></codedpdisoci></codedpdisoci></codedpdisoci></codedpdisoci></codedpdisoci></codedpdisoci></pre>			0
PartialUpdateExample PartialUpdateExample : example for Waveshare 1.54°, 2.31° and 2.9° e-Paper and the same e-papers from Dalian Good Display Inc. I The e-paper displays are available from: https://www.aliexpress.com/store/product/Aholesale-1.54inch-F-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 http://www.aliexpress.com/store/product/ProductAnth-2007_3835&product_id=55120 or https://www.aliexpress.com/store/product/2001_54-inch-partial-refresh-Small-size-dot-matrix-e-paper-display/600281_32815089163.html Supporting Arduino Forum Topics: Waveshare e-paper displays with SPI: http://forum.arduino.cc/index.php?topic=487007.0 Good Dispay dPaper for Maveshare 2.9inch e-Paper to generic ESP266 BUSY > DC. RST > DA, DC > DS, CS > DB, CK > DS, DT > DT, MD > MD > MD > MD > MD > 3.3V > 3.3V mapping suggestion from Waveshare 2.9inch e-Paper to generic ESP266 BUSY > DC. RST > DA, DC > DS, CS > DB, CK > DS, DT > DF, MD > MD > MD > MD > MD > MD > 3.3V > 3.3V mapping suggestion for Maveshare 2.9inch e-Paper to generic ESP266 BUSY > DC. RST > DA, DC > DS, CS > SS (CK > DS, DT > DF, MD > MD > MD > MD > MD > MD > 3.3V > 3.3V mapping suggestion for Maveshare 2.9inch e-Paper to generic ESP266 BUSY > DC. RST > D, DC > T, CS > SS(S), CLK > SC K(18), DIN > OFIDI3, MD > GND, 3.3V > 3.3V mapping suggestion for MAVESHARE 2.9inch e-Raper to generic ESP266 BUSY > 2, RST > 10, DC > 17, CS > SS(S), CLK > SC K(18), DIN > MOSI(23), GND > GND, 3.3V > 3.3V mapping suggestion for MAVESHARE 2.9inch e-Raper to generic ESP266 BUSY > 2, RST > 10, DC > 17, CS > SS(S), CLK > SC K(18), DIN > MOSI(23), GND > GND, 3.3V > 3.3V mapping suggestion for MAVE, MON, NANO etc. BUSY > 7, RST > 9, DC > 8, CS > 10, CLK > 13, DIN > 11 include deception. select the display class to use, only one include deception. include deception. select the display(GAUSASTIZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/GAUDEWAZIZZ/			
<pre>initizes/conductangles = example for marginger 1/3, first min is grapher and the same expansion for ballan bood bisplay for COEPDISC1. [] for earced by Jean-Marc Zingg based on demo code from Good Display for COEPDISC1. [] the e-paper displays are available from: thtps://www.aliexpress.com/store/product/product&path=2807 8363&product_id=35120 or https://www.aliexpress.com/store/product/product&path=2807 8363&product_id=35120 or https://www.aliexpress.com/store/product/forum.arduino.cc/index.php?topic=487007.0 Woweshare = opaper displays with SPI: http://forum.arduino.cc/index.php?topic=487007.0 Good Dispay ePaper for Arduino : https://forum.arduino.cc/index.php?topic=487007.0 Http://www.shiresels.com/store/product.com/shireself to Weeso DI min Http://www.shiresels.com/shireself to Weeso DI min Http://wweesoff</pre>	artialUpdateExample	av Inc	
<pre>treated by Jean-Ard 2.htg Daace on demo code from Good Liplay for GePrUSUL. The e-paper displays are available from: https://www.aliexpress.com/store/product/Anholesale-1-54inch-E-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 https://www.aliexpress.com/store/product/Epoth-E-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 or https://www.aliexpress.com/store/product/Epoth-E-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 or https://www.aliexpress.com/store/product/Epoth-E-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 or https://www.aliexpress.com/store/product/Epoth-E-Ink-display-Module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 fully-interface-Support displays with SPI http://forum.arduino.cc/index.php?topic=487007.0 Good Dispay effect for Muno: https://forum.arduino.cc/index.php?topic=487007.0 Good Dispay effect for Muno: https://forum.arduino.cc/index.php?topic=487007.0 BoS' or 20, RST > 00, C > 00, C > 00, C > 00, C > 00, 3.3V > 3.3V mapping suggestion for Muno.With offering first for MUN > MOSI(23), GND > GND , 3.3V > 3.3V mapping suggestion for Amy. Muno.Muno BuS' or 2, RST > 0, D C > 8, C C > 10, C K > 3, DIN > 11 include -GoodDispay.composition for M</pre>	ranciacupuarezzampre , exampre for mavesnore 1.34 , 2.31 and 2.5 enaper and the same e-papers from bactan dood bispr	T T	
<pre>/ The e-paper displays are available from: https://www.aliexpress.com/store/product/Aholesale-1-54inch-E-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21023 http://www.aliexpress.com/store/product/Ed021-54-inch-partial-refresh-Small-size-dot-matrix-e-paper-display/600281_32815089163.html / supporting Arduino Forum Topics: //www.aliexpress.com/store/product/forum.arduino.cc/index.php2topic=482007.0 Good Dispay ePaper for Arduino : https://forum.arduino.cc/index.php2topic=482007.0 Good Dispay ePaper for Arduino : https://forum.arduino.cc/index.php2topic=480411.0 Http://www.aliexpress.com/store/product/Spinte-Spinte-480007.0 Good Dispay ePaper for Arduino : https://forum.arduino.cc/index.php2topic=480411.0 Https://www.aliexpress.com/store/product/Spinte-Spinte-480007.0 Good Dispay ePaper for Maveshare 2.0inch e-Paper to Weeos DI mini BUSY ~ 902, RST ~ D4, DC ~ 03, CS ~ 080, CLK ~ 05, DIN ~ D7, GND ~ 0HD, 3.3V ~ 3.3V mapping suggestion from Waveshare 2.0inch e-Paper to generic ESP266 WISY ~ 0ftps T ~ 16, DC ~ 17, CS ~ 55(C), CLK ~ 05 F0104, DIN ~ 0601013, GND ~ 0HD, 3.3V ~ 3.3V mapping suggestion for ESP32, e.g. LOLIN32, see/variants//pins_arduino.h for your board MOTE: there are variants with different pins for SFI ! CHECK SFI PINS 0F YOUR BOARD WISY ~ 4, RST ~ 9, D, CC ~ 0, S, CS ~ 10, CLK ~ 35(C), LN ~ 9 SCI(12), DIN ~ 9 GND, 3.3V ~ 3.3V mapping suggestion for AVR, UNO, NANO etc. BUSY ~ 7, RST ~ 9, D, CC ~ 8, CS - 10, CLK ~ 31, DIN ~ 11 // include Library, include base class, make path known mclude <codefd0150c 1.54*="" b="" cocgpdejsci1.copp="" w<br="">#include <codefd0150c 1.54*="" b="" cocgpdejsci1.copp="" w<br="">#include <codefd02183c10pe 2.3*="" cag0ed3283.copp="" d="" w<br="">#include <codefd02383c00fd021283.copp 2.3*="" d="" w<br="">#include <codefd02383c00fd023283c0pe 2.3*="" d="" w<br="">#include <codefd023283c00fd0232373z 2.4*="" b<="" copp="" td=""><td>created by Jean-Marc Zingg based on demo code from Good Display for GDEP015UCI.</td><td>1</td><td></td></codefd023283c00fd0232373z></codefd02383c00fd023283c0pe></codefd02383c00fd021283.copp></codefd02183c10pe></codefd02183c10pe></codefd02183c10pe></codefd02183c10pe></codefd0150c></codefd0150c></pre>	created by Jean-Marc Zingg based on demo code from Good Display for GDEP015UCI.	1	
<pre>https://www.aliexpress.com/store/product/pholesale-1-54inch-E-Ink-display-module-with-embedded-controller-200x200-Communicate-via-SPI-interface-Supports/21623 http://www.bau/ed.com/index.php?route-product/productSpath=2807_8365&product_id=35120 or https://www.aliexpress.com/store/productStore/produ</pre>	The e-paper displays are available from:		
<pre>/ http://www.bluy-icd.cow/index.php?route-product/E001-154-inch-partial-refresh-Small-size-dot-matrix-e-paper-display/600281_32815089163.html / supporting Arduino Forum Topics: //waveshare e.paper displays with SPI: http://forum.arduino.cc/index.php?topic=487007.0 / Good Dispay Paper for Arduino : https://forum.arduino.cc/index.php?topic=487007.0 // Good Dispay Paper for Arduino : https://forum.arduino.cc/index.php?topic=487007.0 // BuSira Set Set Set Set Set Set Set Set Set Set</pre>	nttps://www.aliexpress.com/store/product/Wholesale-1-54inch-E-Ink-display-module-with-embedded-controller-200x200-Commu	inicate-via-SPI-interface-S	upports/21623
<pre>/ upporting Arduino Forum Topics: /www.share e.paper displays with SPI: htp://forum.arduino.cc/index.php?topic=497007.0 Good Dispay ePaper for Arduino : https://forum.arduino.cc/index.php?topic=436411.0 / mapping suggestion from Waveshare 2.9inch e-Paper to Weeos D1 mini / BuSY > 02, BST > 04, DC > 08, CS > 06, CK >> 05, DN >> 07, GHD >> 0HD, 3.3V >> 3.3V / mapping suggestion from Waveshare 2.9inch e-Paper to generic ESP8266 BuSY >> OF104, RST >> OF102, DC >> OF100, CS >> OF1015, CLK >> OF1014, DIN >> OF013, GND >> GND, 3.3V >> 3.3V / mapping suggestion for Maveshare 2.9inch e-Paper to generic ESP8266 BuSY >> OF104, RST >> OF102, DC >> OF100, CS >> OF1015, CLK >> OF1014, DIN >> OF013, GND >> GND, 3.3V >> 3.3V / mapping suggestion for ESP32, e.g., LOLINS2, see, /variants//pins Grivour board MOT: there are variants with different pins for SP1 1 CHECK SPI PINS GF YOUR BOARD BUSY >> 4, RST >> 0, DC >> 17, CS >> SS(), CLK >> SK(10), DIN >> MOSI(23), GHD >> GND, 3.3V >> 3.3V / mapping suggestion for ARM, UNO, NANO etc. / BUSY >> 7, RST >> 0, DC >> 8, CS >> 10, CLK >> 13, DIN >> 11 / include Library, include base class, make path known mclude =</pre>	<u>attp://www.buy-lcd.com/index.php?route=product/product&path=2097_8363&product_id=35120</u> or https://www.aluexpress.com/store/product/E001-1-54-inch-partial-refresh-Small-size-dot-matrix-e-paper-display/600281	_32815089163.html	
<pre>/ mapping suggestion from Waveshare 2.9inch e-Paper to Wemos DI mini BUSY > 02, RST >> D4, DC -> D3, CS >> D8, CLK >> D5, DIN >> D7, OHD >> OHD, 3.3V >> 3.3V / mapping suggestion from Waveshare 2.9inch e-Paper to generic ESP8266 BUSY >> OF(D4, RST >> OF(D2, DC >> OF(D0, CS >> OF(D14, DIN >> OF(D13, GND >> GND, 3.3V >> 3.3V / mapping suggestion for ESP22, e.g. LOLIN22, see/Variants//pins arduine.h for your board / MDT: there are variants with different pins for SP1 (CHCK SPI PINS OF YOUR BOARD / BUSY >> 4.RC >> 17, CS >> 55(5), CLK >> 55(CLK >> 55(CLK >> 55(CLK >> 6F(D14, DIN >> MOSI(23), GND >> GND, 3.3V >> 3.3V / mapping suggestion for AVR, UNO, NANO etc. / BUSY >> 4, RST >> 10, DC >> 17, CS >> 55(5), CLK >> 13, DIN >> 11 / include tibrary, include base class, make path known include <<kcepd1>DC /> 12, See a. (J. S.4* bAve #include <<kcepd1>DC /> 13, SAV >> 13, DIN >> 11 / include tibrary, include base class, make path known include <<kcepd1>DC // 2.3* b/w #include <<kcepd1>DC // 2.5* b/w #include </kcepd1>DC // 2.5* b/w #include </kcepd1>DC // 2.5* b/w</kcepd1></kcepd1></kcepd1></kcepd1></kcepd1></kcepd1></pre>	Supporting Arduino Forum Topics: Waveshare e-paper displays with SPI: <u>http://forum.arduino.cc/index.php?topic=487007.0</u> Good Dispay ePaper for Arduino : <u>https://forum.arduino.cc/index.php?topic=486411.0</u>		
<pre>/ sapping suggestion from Waveshare 2. Joinch e-Paper to generic ESP2266 / BUSY > GPI04, RST >> GPI02, DC >> GPI00, CC >> GPI015, CLK >> GPI014, DIN >> GPI013, GND >> GND, 3.3V >> 3.3V / apping suggestion for ESP32, e.g., LOLIN32, see/variants//pins.arduino.h for your board / MOTE: there are variants with different pins for SPI ! CHECK SPI PINS OF YOUR BOARD / WOTY >> 4, RST >> 10, DC >> 11, CS >> SS(5), CLK >> SS(10), DIN >> MOSI(23), GND >> GND, 3.3V >> 3.3V / mapping suggestion for AVR, UNO, NANO etc. * BUSY >> 7, RST >> 9, DC >> 8, CS >> 10, CLK >> 13, DIN >> 11 / include library, include base class, make path known mclude = GACEPDISOCI.copp // 1.54* b/W #include = GACEPDISOCI.copp // 1.54* b/W #include = GACEPDISOCI.copp // 2.3* b/W #include = GACEPDISOCI.copp // 2.3* b/W #include = GACEPDISOCI.copp // 2.4* b/W #inclu</pre>	mapping suggestion from Waveshare 2.9inch e-Paper to Wemos D1 mini BUSY -> D2, RST -> D4, DC -> D3, CS -> D8, CLK -> D5, DIN -> D7, GMD -> GMD, 3.3V -> 3.3V		
<pre>/ mapping suggestion for ESP32, e.g. LOLIN32, see/variants//pins arduino.h for your board / WDF: three are variants with different pins for SPI L LOEK SPI PINS for FVOWE BOARD / BUSY ~> 4, RST ~> 16, DC ~> 17, CS ~> SS(5), CLK ~> SCK(18), DIN ~> MOSI(23), GND ~> GND, 3.3V ~> 3.3V / mapping suggestion for AVR, UNO, NANO etc. / BUSY ~> 7, RST ~> 9, DC ~> 8, CS ~> 10, CLK ~> 13, DIN ~> 11 / include Library, include base class, make path known include <gkepdisol gkoeppisoll.cpp=""> // 1.54* b/w frinclude <gkepdisol gkoeppisoll.cpp=""> // 1.54* b/w frinclude <gkepdisol gkoeppisoll.cpp=""> // 2.3* b/w frinclude <gkepdisol gkoeppisoll.cpp=""> // 2.3* b/w frinclude <gkepdisol gkoeppisoll.cpp=""> // 2.4* b/w frinclude <gkepolysak dengersia.cpp=""> // 2.4* b/w/ frinclude <gkepolysak dengersia.cpp=""> // 2.5* b/w frinclude <gkepolysak deng<="" dengersiak="" td=""><td>mapping suggestion from Waveshare 2.9inch e-Paper to generic ESP8266 BUSY -> GPI04, RST -> GPI02, DC -> GPI00, CS -> GPI015, CLK -> GPI014, DIN -> GPI013, GND -> GND, 3.3V -> 3.3V</td><td></td><td></td></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepolysak></gkepdisol></gkepdisol></gkepdisol></gkepdisol></gkepdisol></pre>	mapping suggestion from Waveshare 2.9inch e-Paper to generic ESP8266 BUSY -> GPI04, RST -> GPI02, DC -> GPI00, CS -> GPI015, CLK -> GPI014, DIN -> GPI013, GND -> GND, 3.3V -> 3.3V		
/ apping suggestion for AVR. UMO. NAMO etc. / BUSY >> 7, RST >> 9, DC >> 8, CS >> 10, CLK >> 13, DIN -> 11 / include library, include base class, make path known include <6xEPD.h> / select the display class to use, only one // include <5xCBPD150C1/cK0EPD150C1.cpp> // 1.54* b/w // include <5xCBPD150C1/cK0EPD150C1.cpp> // 2.3* b/w // include <5xCBP045201.ccSEPD12131.cpp> // 2.3* b/w // include <5xCBP045201.ccSEPD3201.cpp> // 2.2* b/w // these display do not fully support partial update // these display do not fully support partial update	amping suggestion for ESP32, e.g. LOLIN32, see/variants//pins.arduino.h for your board MDTE: there are variants with different pins for SPI (+ CHCK SPI PINS oF YOUR BOARD BUSY → 4, RST → 16, DC → 17, CS → SS(5), CLK → SCK(18), DIN → MOSI(23), GND → GND, 3.3V → 3.3V		
/ include library, include base class, make path known include GAGEPD.h> ' select the display class to use, only one #include GAGEPDISO(1/GAGEPDISO(1,cpp> // 1.54* b/w #include GAGEPDISO(1/GAGEPDISO(1,cpp> // 1.54* b/w #include GAGEPDISO(1/GAGEPDISO(1,cpp> // 2.5* b/w #include GAGEPDISO(1/GAGEPDISO(1,cpp> // 2.2* b/w #include GAGEPDISO(1/GAGEPDISO(1,cpp> // 2.2* b/w these displays do not fully support partial update #include GAGEPDISO(1/GAGEPDISO(1,cpp> // 1.54* b/w/c 150×157 #include GaGEPDISO(1/GAGEPDISO(1,cpp> // 2.5* b/w) #include GAGEPDISO(1,cpp> // 2.5* b/w) #include GAGEPDISO(1,cpp	mapping suggestion for AVR, UNO, NANO etc. BUSY -> 7, RST -> 9, DC -> 8, CS-> 10, CLK -> 13, DIN -> 11		
/ yelact the display class to use, only one 'finclude <scopedisol 1.54'="" b="" scopedisol(.cpp)="" w<br="">'finclude <scopedisol 2.3'="" b="" scopedisol(.cpp)="" w<br="">'finclude <scopedisol 2.9'="" b="" scopedisol(.cpp)="" w<br="">'finclude <scopedisol 2.2'="" b="" scopedisol(.cpp)="" w<br="">'finclude <scopedisol(scopedisol(.cpp) 2.2'="" b="" w<br="">'finclude <scopedisol(.cpp) 2.2'="" b="" w<br="">'finclude <scopedisol(< td=""><td>include library, include base class, make path known <mark>clude <gxepd.h></gxepd.h></mark></td><td></td><td></td></scopedisol(<></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol(scopedisol(.cpp)></scopedisol></scopedisol></scopedisol></scopedisol>	include library, include base class, make path known <mark>clude <gxepd.h></gxepd.h></mark>		
ш —	select the display class to use, only one include < <pre>cxx0DF0150C1/Cxx0DF0150C1.cpp> // 1.54* b/w include <<xx0df023b1 cxx0df023b1.cpp=""> // 2.13* b/w include <<xx0df023b1 cxx0df023b1.cpp=""> // 2.9* b/w include <<xx0df04271 cxx0df04272.cpp=""> // 4.2* b/w these displays do not fully support partial update these displays do not fully support partial update</xx0df04271></xx0df023b1></xx0df023b1></pre>		
			>
		Nodo22c 20MU-031CO	in an Maulthul ISBO
		Node32s, 80MHz, 92160	0 on /dev/ttyUSB0

rst:0x1 (POWERON RESET),boot:0x13 (SPI FAST FLASH BOOT) configsip: 0, SPIWP:0xee clk drv:0x00,g drv:0x00,d drv:0x00,cs0 drv:0x00,hd drv:0x00,wp drv:0x00 de:DIO, clock div:2 pad:0x3fff0018,len:4 ad:0x3fff001c,len:4732 ad:0x40078000,len:7496 ad:0x40080400.len:5512 try 0x4008114c (388) cpu start: Pro cpu up. (389) heap init: Initializing. RAM available for dynamic allocation: (392) heap init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM (398) heap init: At 3FFC4F48 len 0001B0B8 (108 KiB): DRAM (405) heap init: At 3FFE0440 len 00003BC0 (14 KiB): D/IRAM (411) heap init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM (417) heap init: At 40091448 len 0000EBB8 (58 KiB): IRAM (424) cpu start: Pro cpu start user code (218) cpu start: Starting scheduler on PRO CPU. tting up LEDs tting up Buttons tting up Sensor I2C tting up BME280 tting up TSL2591 e values[0]: 2172 - 21.72C values[1]: 25929420 - 1012.86hPa values[2]: 44558 - 43.51% values[0]: 48 values[1]: 21 Good itialize the Board LED as a PWM... Success break hit <ctrl>+c then enter: breathTimer.deinit() Error: [Errno 2] ENOENT croPython v1.9.4-560-g185716514 on 2018-09-20; ESP32 module with ESP32 Type "help()" for more information. >>>

Why is this different?

- Quick, iterative, development
- . Most of the advantages of Python
- 0 to blinking LED very quick
- Mostly no need to compile anything
- Lots of default functionality, and upip (library / package management!)

Why is this possible?

- Same reason IoT is becoming ubiquitous
 - $\circ~$ MCUs & CPUs are getting more powerful, and cheaper
- ESP32 on the SensorNode cost \$5.10 to place on the board.
 - Dual Core
 - Wifi (802.11b/g/n up to 150Mbps 2.4GHz)
 - Bluetooth (v4.2 BR/EDR & BLE)
 - 4MB of flash
 - 520KB RAM
- There's lots of competition in this space



Flashing MicroPython:

With the VM:

- Select the VM, plug in SensorNode
 - Should cause it to attach to the VM, if it's not VM -> Removable Devices and attach it
- Helper script (specific to this tutorial)

flash_sensornode.sh

- Sets Serial port (usually /dev/ttyUSB0)
- Fully erases the flash on the ESP32
 - esptool.py --chip esp32 --port
 "\${USBPORT}" erase_flash
- Flashes MicroPython
 - esptool.py --chip esp32 \
 --port "\${USBPORT}" --baud 460800 \
 write_flash -z 0x1000 "\${flash_file}"

Without the VM:

- Serial Drivers
 - Linux: Driver in Most Distros
 - Windows / Mac: Install Silicon Mechanics CP2104 <u>https://www.silabs.com/products/development-to</u> ols/software/usb-to-uart-bridge-vcp-drivers
- Download / Install esptool
 - This requires Python
 - Linux:

distro packages are available

• Windows / Mac:

use pypi to install

- Download MicroPython & Upload it to the board
 - O <u>http://micropython.org/download#esp32</u>
 - esptool.py --chip esp32 \
 --port /dev/ttyUSB0 erase_flash && \
 esptool.py --chip esp32 --port \
 /dev/ttyUSB0 write_flash -z 0x1000 \
 <path to micropython .bin>

Make Sure the SensorNode is 'on'

Blinking Charge Indicator





Helpful tip:

If there's a flashing light on the board it's on (it's the charging indicator light). If it's solid, it's off.

The switch is on the side with the USB port:

- Down = On
- Up = Off

Breaking down the flash commands

esptool.py \ --chip esp32 \ --port /dev/ttyUSB0 \ erase_flash \ 8& \ esptool.py \ --chip esp32 \ --port /dev/ttyUSB0 \ write_flash \ -z 0x1000 \ <path to micropython .bin>

Identifies which chip variant we are dealing with# Identifies which port the serial device is on# Erases the flash area of the chip

(not including the boot loader area)

Identifies which chip variant we are dealing with# Identifies which port the serial device is on# Indicates to write to the flash chip

- # Indicates WHERE on the flash chip to write to
- # What to flash to the chip

What this should look like:

[root@tutorial-base ~]# dmesg | tail -n 8

[...]

[86344.904683] cp210x 2-2.1:1.0: cp210x converter detected [86344.915286] usb 2-2.1: cp210x converter now attached to ttyUSB0

[root@tutorial-base ~]# ./flash_sensornode.sh Flash File: esp32-20190214-v1.10-98-g4daee3170.bin esptool.py v2.7-dev

Serial port /dev/ttyUSB0

Connecting.....

Chip is ESP32D0WDQ6 (revision 1)

Features: WiFi, BT, Dual Core, Coding Scheme None

MAC: 30:ae:a4:86:c7:64

Uploading stub...

Running stub...

Stub running...

Erasing flash (this may take a while)...

Chip erase completed successfully in 4.4s Hard resetting via RTS pin... esptool.pv v2.7-dev Serial port /dev/ttyUSB0 Connecting..... Chip is ESP32D0WDQ6 (revision 1) Features: WiFi, BT, Dual Core, Coding Scheme None MAC: 30:ae:a4:86:c7:64 Uploading stub... Running stub... Stub running... Changing baud rate to 460800 Changed. Configuring flash size... Auto-detected Flash size: 4MB Compressed 1133232 bytes to 714809... Wrote 1133232 bytes (714809 compressed) at 0x00001000 in 18.6 seconds (effective 488.0 kbit/s)... Hash of data verified

Leaving... Hard resetting via RTS pin... [root@tutorial-base ~]#

Open up the serial console

- Minicom:
 - minicom -D /dev/ttyUSB0 --baudrate 115200 (to exit <ctrl>c-q)
- Screen:
 - screen /dev/ttyUSB0 115200n8 (to exit <ctrl>c-A \)
- Windows: use PuTTY
Reset the board



On the serial console...

ets Jun 8 2016 00:22:57

rst:0x1 (POWERON RESET),boot:0x13 (SPI FAST FLASH BOOT) configsip: 0, SPIWP:0xee clk drv:0x00,q drv:0x00,d drv:0x00,cs0 drv:0x00,hd drv:0x00,wp drv:0x00 mode:DIO, clock div:2 load:0x3fff0018,len:4 load:0x3fff001c.len:5060 load:0x40078000.len:8788 ho 0 tail 12 room 4 load:0x40080400.len:6772 entry 0x40081610 (428) cpu start: Pro cpu up. (428) cpu start: Application information: (428) cpu_start: Compile time: 12:32:34 (430) cpu start: Compile date: Feb 14 2019 (436) cpu_start: ESP-IDF: v3.3-beta1-268-g5c88c5996 (442) cpu_start: Single core mode (447) heap init: Initializing. RAM available for dynamic allocation: (454) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM (460) heap_init: At 3FFB92B0 len 00026D50 (155 KiB): DRAM I (466) heap_init: At 3FFE0440 len 0001FBC0 (126 KiB): D/IRAM (472) heap init: At 40078000 len 00008000 (32 KiB): IRAM (479) heap init: At 40092834 len 0000D7CC (53 KiB): IRAM I (485) cpu_start: Pro cpu start user code (55) cpu start: Starting scheduler on PRO CPU. OSError: [Errno 2] ENOENT MicroPython v1.10-98-g4daee3170 on 2019-02-14; ESP32 module with ESP32 Type "help()" for more information. >>>

Quick Hello World!

>>> print("Hello World!") Hello World!

>>>

Now to Blink an LED!

>>> import machine

>>> led_pin = machine.Pin(0, machine.Pin.OUT)

>>> led_pin.on()

>>> led_pin.off()



Note: You'll quickly find the on() turns the LED off, and off() turns the LED on. To "Fix"

>>> led = machine.Signal(led_pin, invert=True)
>>> led.off()
>>> led.on()

Some interesting things to note

- boot.py
 - executed on every start, good for setting up the board (good place for wifi settings for example)
- main.py
 - Run after boot.py, think of it like the autoexec.bat
- It's possible to upload more files to the board
 - Ampy <u>https://github.com/adafruit/ampy</u>
- Tab completion works in the repl prompt
- <ctrl>+e at the repl prompt puts you into "paste" mode

Disconnect From Serial before trying file transfers!

- Minicom:
 - to exit: <*ctrl*>*c*-*q*
- Screen:
 - to exit: <ctrl>c-A \ y
- Putty:
 - Hit the X and close the application

Where to go from here

Setup Wifi in client mode

- ampy --port /dev/ttyUSB0 get boot.py | tee boot.py
 - # This file is executed on every boot (including wake-boot from deepsleep) #import esp
 - #esp.osdebug(None)
 - #import webrepl
 - #webrepl.start()
- Add to boot.py:
 - # This file is executed on every boot (including wake-boot from deepsleep)
 #import esp
 #esp.osdebug(None)
 #import webrepl
 #webrepl.start()
 import network
 sta = network.WLAN(network.STA_IF)
 sta.active(True)
 sta.connect("ALE", "Penguins")
- ampy --port /dev/ttyUSB0 put boot.py

Re-connect to Serial and check:

- >>> sta.ifconfig()
 ('192.168.123.456', '255.255.255.0', '192.168.123.1', '192.168.123.1')
 >>> sta.status()
 1010
 >>> sta.isconnected()
 True
 >>>

```
- >>> import socket
```

```
>>> addr_info = socket.getaddrinfo("towel.blinkenlights.nl", 23)
>>> addr = addr_info[0][-1]
>>> s = socket.socket()
>>> s.connect(addr)
>>> while True:
... data = s.recv(500)
... print(str(data, 'utf8'), end=")
...
<ctrl>+c will stop the while loop
```

One more thing to note, but not try here...

- Access Point Mode (can be used with client mode at the same time, albeit slowly)
 - >>> ap = network.WLAN(network.AP_IF)
 >>> ap.active(True)
 >>> #ap.config(essid="network-name", authmode=network.AUTH_WPA_WPA2_PSK, password="abcdabcdabcd")
 - Can be added to boot.py, same as the client information

Links to more resources

- <u>https://github.com/unreproducible/tinysnakes</u>
- <u>https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html</u> (note: most of the ideas are the same, the boards ARE different)
- <u>https://boneskull.com/micropython-on-esp32-part-1/</u>
- https://www.cnx-software.com/2017/10/16/esp32-micropython-tutorials/

• Any questions before you start this on your own?

John 'Warthog9' Hawley | <u>warthog9@eaglescrag.net</u> | @warty9

IoT-ALE: Reading Sensor Data with I2C

Jon Mason

SCaLE 17x - March 2019

I2C Some background

- Released in 1982
- Bus Protocol
- Devices use addresses
- 2-pins needed
 - Clock (scl)
 - Data (sda)
- SCL & SDA pulled-up against voltage to ship (Vdd)
 - Level shifting can be complicated to get right
- Upwards of 3.4Mbps
 - More realistically: ~1Mbps
 - Most devices communicate in Kbps
- SMBus is derived from, but not identical to, I2C
 - Devices may claim to be one or the other not really consistent



What this looks like on the bus:



- 1. Data transfer is initiated with a *start* bit (S) signaled by SDA being pulled low while SCL stays high.
- 2. SCL is pulled low, and SDA sets the first data bit level while keeping SCL low (during blue bar time).
- 3. The data are sampled (received) when SCL rises for the first bit (B1). For a bit to be valid, SDA must not change between a rising edge of SCL and the subsequent falling edge (the entire green bar time).
- 4. This process repeats, SDA transitioning while SCL is low, and the data being read while SCL is high (B2, ...Bn).
- 5. The final bit is followed by a clock pulse, during which SDA is pulled low in preparation for the *stop* bit.
- 6. A stop bit (P) is signaled when SCL rises, followed by SDA rising.

Addresses

- Device specific implementation
- Some devices only provide a single address
 - One device per-bus
- Address on the bus "needs" to be unique
- 7-bit address normal
 - 128 Devices normally
 - 10-bit exists, very little uses it
 - 10-bit gives you 1008 devices (reserved addresses)



```
I2Cdetect (Linux)
```

Devices at: • 0x56

• 0x68

- I2C is *NOT* discoverable, detection is not guaranteed
- Random probing can cause systems to crash you are warned

SensorNode has 2 x I2C devices:

BME280

- Temperature
- Humidity
- Relative Pressure

TSL2591

- Full Spectrum Light Sensor
- IR Spectrum Light Sensor



Figuring out Address - See the Schematic(s)

https://github.com/unreproducible/sensornode/blob/master/Schematic%20-%20sensornode.pdf

BME280:



TSL2591



Ordering Code	Address	Interface	
TSL25911FN	0x29	I ² C V _{bus} = V _{DD} Interface	
TSL25913FN*	0x29	$I^2 C V_{bus} = 1.8 V$	



Time to read some data

- 1. Exit screen
- 2. Upload the following using ampy: # ampy --port /dev/ttyUSB0 put sensornode-stuff/src/bme280.py
 # ampy --port /dev/ttyUSB0 put sensornode-stuff/src/tsl2591.py
 # ampy --port /dev/ttyUSB0 put sensornode-stuff/src/usmbus
 Note the last one is a directory
- 3. Open up the serial port again

Confirm file upload

```
>>> import os
>>> os.listdir()
['boot.py', 'bme280.py', 'tsl2591.py', 'usmbus']
>>>
```

BME280 - Environment Sensor

>>> from machine import Pin, I2C
>>> import machine
>>> import bme280

>>> pin_i2c_scl = 22 >>> pin_i2c_sda = 21

>>> bme280_address = 0x77

>>> sensor_i2c = I2C(scl=Pin(pin_i2c_scl), sda=Pin(pin_i2c_sda))

>>> bme = bme280.BME280(i2c=sensor_i2c, address=bme280_address)

>>> bme.values ('26.84C', '1015.59hPa', '17.71%') >>>



TSL2591 - Light Sensor

>>> import tsl2591
>>> tsl = tsl2591.Tsl2591()
>>> tsl.get_full_luminosity()
(58, 14)
>>>



The TSL2591 driver is a very setup than the BME280. The I2C bus, and address, are hard Coded into the driver:

```
55def __init__(self, scl_pinno=22, sda_pinno=21):56self.i2c = I2C(scl=Pin(scl_pinno, Pin.IN),57sda=Pin(sda_pinno, Pin.IN))
```

It also makes use of more SMBus like support (usmbus)

Places to find more information on I2C:

- <u>https://i2c.info/</u>
- <u>https://en.wikipedia.org/wiki/I%C2%B2C</u>
- <u>https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS00</u>
 <u>2.pdf</u>
- <u>https://cdn-shop.adafruit.com/datasheets/TSL25911_Datasheet_EN_v1.pdf</u>

IoT-ALE: Reading and Writing to SPI SDcards

Nisha Kumar

SCaLE 17x - March 2019

SPI Background

- Not a hard defined standard like I2C
 - Ubiquitous despite no hard standard
 - Data on the bus is effectively device unique
 - Quad SPI can add 2 more data lines, uncommonly used
- Faster than I2C
 - Possible to go >10Mbps
- Duplex communications
 - Master Out Slave In (MOSI)
 - Master In Slave Out (MISO)
- Hardwired device selection



Where this gets messy...

- While fast, it's not easy to implement
- Chip select lines can get
 very expensive, very quickly
- Some devices need more than the minimum 4* wires





* Minimum is based on duplex operation, some devices are write or read only and you only need 3 wires then

SPI Screens, cases in point as "odd"

E-link

- SPI Like Interface
- Busy pin
- Reset pin
- Data/Command (DC) pin
- Write-only device (MOSI)
- 8-pins (including Vcc & GND)

OLED Screen

- SPI Like interface
- Write-only device (MOSI)
- Reset pin
- Data/Command (DC) pin
- 7-pins (including Vcc & GND)





Normal SPI Device

BME280 (SPI mode)

- CSB Chip Select
- SCL Clock
- SDA MOSI (serial data in)
- SDO MISO (serial data out)
- GND Ground
- VCC Power



I2C

UCC

GND

CSB

SDcards and SPI

- SDcards have two basic modes:
 - SD mode
 - \circ SPI mode
- SPI mode disadvantages:
 - Slower transfers (no parallel data)
 - 'U' modes aren't supported
- SPI mode advantages:
 - Easier to implement
 - Less hardware needed
 - Simpler interface



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	Х
9	DAT2	х



Pin	SD	SPI
1	DAT2	Х
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	Х

Hardware vs. Software Implementation

Hardware:

- 4 SPI Busses
 - SPI0 typically dedicated to Flash
 - SPI1 tied to same pins as SPI0
 - HSPI (SPI2)
 - CS: 15
 - SCLK: 14
 - MISO: 12
 - MOSI: 13
 - QUADWP: 2
 QUADHD: 4
 - VSPI (SPI3)
 - CS: 5
 - SCLK: 18
 - MISO: 19
 - MOSI: 23
 - QUADWP: 22
 - QUADHD: 21

Software

- Any pins will do
- Bitbanged in software / timers
- SensorNode uses:
 - CS: 15 • SCLK: 14
 - SCLK: 14 • MISO: 12
 - MISO: 12
 MOSI: 13
 - QUADWP: -
 - QUADHD: -

Wiring up an SDcard to an MCU

_Micro SD Card Cage





GND

Prep work for using the SDcard

- 1. Exit screen
- 2. Upload the following using ampy:# ampy --port /dev/ttyUSB0 put sensornode-stuff/src/sdcard.py
- 3. Open up the serial port again

Lets look at some code - Setup the SPI Interface

Software (use this on SensorNode)

>>> from machine import Pin, SPI
>>> cs = Pin(15, Pin.OUT)
>>> mosi = Pin(13, Pin.OUT)
>>> miso = Pin(12, Pin.IN)
>>> sck = Pin(14, Pin.OUT)
>>> spi_bus = SPI(sck = sck,
mosi = mosi, miso = miso)

Hardware (for comparison only) >>> from machine import Pin, SPI >>> cs = Pin(15, Pin.OUT) >>> spi_bus = SPI(2) Adding the SD card to the mix

- 1. Plug in the SD card
 - SD Card is on the back behind the buttons
- 2. Add the following:
- >>> import sdcard
 >>> sd = sdcard.SDCard(spi_bus, cs)
 >>>



What this looks like, without the SD card in place: >>> sd = sdcard.SDCard(spi_bus, cs) Traceback (most recent call last): File "<stdin>", line 1, in <module> File "sdcard.py", line 54, in __init__ File "sdcard.py", line 82, in init_card OSError: no SD card >>>

Mounting the SDCard

• You mount it to the filesystem like Unix / Linux

```
    >>> import os
    >>> os.mount(sd, '/sd')
    >>> os.listdir('/')
    ['sd', 'boot.py', 'bme280.py', 'sdcard.py', 'tsl2591.py', 'usmbus']
    >>> os.listdir('/sd')
    ['MISC', 'DCIM', 'old']
    Contents here will likely be empty unless you've Put things on the card already
```

Reading & Writing to the SD card

```
>>> f = open("/sd/demofile.txt", "a")
>>> f.write("Hello World!")
12
>>> f.close()
>>> f = open("/sd/demofile.txt", "r")
>>> f.read()
'Hello World!'
>>>
```
IoT-ALE:

Connecting to the Internet MQTT

putting the I in IoT

John 'Warthog9' Hawley

SCaLE 17x - March 2019

Let us lay some ground works... What most "home" networks look like:

Firewall

Main Network

Wireless Guest

More Groundwork: IoT devices



A second restriction of the second second

SETUP

MUSEUM

THE LOUVRE MUSEUM



Typical ways devices connect to the Internet

- Through a Gateway:
 - Bluetooth
 - Z-wave
 - o **802.11.6**
 - Zigbee
 - IR
 - Smoke Signals
 - Carrier Pigeons

- Directly:
 - Wifi
 - Ethernet

- Using:
 IPv4
 - o IPv6

Lets come back to this for a minute to talk about IPv4 vs. IPv6

Firewall

Main Network

Wireless Guest

Local Access vs. Remote Access

- IPv4 Local
 - Direct Access
 - Straight Forward
 - Mostly ubiquitous support
- IPv4 Remote
 - NAT traversal
 - Punching holes in firewalls
 - Port Forwarding
 - UPNP
 - Cloud reverse proxies

- IPv6 Local
 - Direct Access
 - Straight Forward
 - Getting more ubiquitous but not there
- IPv6 Remote
 - Direct Access
 - Punching holes in firewalls
 - UPNP
 - Cloud based IP lookup (and/or reverse proxies)

Some general words of caution...

- Think about what you are using the Internet for
- Be mindful of where your services live
- Sometimes UX the user can use may make you less secure
- Always change the default passwords!
- Make it possible to do things without auto-discovery
- Don't always assume you are on the same network as the device
- Upgrade schemes need to be done

Shifting gears & talk about how to talk to the devices

But the real advantage to IoT is the I - Internet!

Lots of good ways to do this...

- MQTT
- Liota
- AMQP
- STOMP
- RabbitMQ
- REST
- WAMP

- ZeroMQ
- Java Message
 - Service (JMS)
- CoAP
- CLOUD!
- XMPP-IOT
- XMPP
 - etc.....



https://xkcd.com/927/ - CC-BY-NC 2.5

Now lets talk about something to try

- MQTT Mosquitto, MQTT broker, good for local passing of data
- Think of it as a message bus on the network
- Clients Subscribe to Topics that can be hierarchical, and listen to the Topic
 - /myhome/groundfloor/livingroom/temperature for example
 - You can listen at any level of the hierarchy, anything below your level will be filtered to you
 - Wildcards, +, are allowed /myhome/+/temperature
- Devices Publish data to topics
 - The data is freeform, the receiving end is expected to interpret it

Lets just try listening...

```
On your laptop/VM:
```

yum install mosquitto

apt-get install mosquitto-clients

then

```
mosquitto_sub \
    -h 10.100.0.5 \
    -t "pugnose/temp/core0" \
    -u "ale" \
    -P "Penguins"
```

Expected output:

What's running on "pugnose":

```
while [[ 1 ]]; do \
     mosquitto pub \
           -h 10.100.0.5 \
           -t "pugnose/temp/core0" \
           -m "$(∖
                 sensors | \
                 grep "Core 0" | \
                 tr " " "\n" | \
                 grep "°" | \
                 head -n 1 \setminus
           ) " \
           -u "ale" \
           -P "Penguins"; \
           sleep 10;\
```

done

+67.0°C

Listening from the IoT device (subscribing)

From the repl prompt:

- >>> from umqtt.simple import MQTTClient
- >>> import socket
- >>> import time
- >>> from ubinascii import hexlify
- >>> CLIENT_ID = hexlify(machine.unique_id())

```
>>> def sub_cb(topic, msg):
```

```
... print((topic, msg))
```

- • •
- •••
- • •

```
>>> c.set callback(sub cb)
```

```
>>> c = MQTTClient(CLIENT_ID,
```

- ... "10.100.0.5")
- >>> c.connect()

```
>>> c.subscribe(b"topic/yourname")
```

- >>> while True: ... if True: ... c.wait_msg() ... else: ... c.check_msg() ... time.sleep(1) ...
- >>> c.disconnect()

From your VM / Laptop

```
mosquitto_pub \
    -h 10.100.0.5 \
    -t "topic/yourname" \
    -m "Hello YourName" \
    -u "ale" \
    -P "Penguins"
```

Publishing from the IoT device

From the repl prompt:

```
>>> from umqtt.simple import MQTTClient
>>> import socket
>>> from ubinascii import hexlify
>>> CLIENT_ID = hexlify(machine.unique_id())
>>> c = MQTTClient(CLIENT_ID,
... "10.100.0.5")
>>> c.connect()
>>> c.publish(b"topic/yourname",
... b"hello from mpy")
>>> c.disconnect()
```

On your laptop/VM:

yum install mosquitto

apt-get install mosquitto-clients

then

```
mosquitto_sub \
    -h 10.100.0.5 \
    -t "topic/yourname" \
    -u "ale" \
    -P "Penguins"
```

For the way advanced!

```
from umgtt.simple import MQTTClient
from machine import Pin
from ubinascii import hexlify
import machine
import micropython
led = Pin(0, Pin.OUT, value=1)
SERVER = "10.100.0.5"
CLIENT ID = hexlify(machine.unique id())
TOPIC = b"topic/yourname"
state = 0
def sub cb(topic, msg):
   global state
   print((topic, msg))
    if msg == b"on":
        led.value(0)
        state = 1
    elif msg == b"off":
        led.value(1)
        state = 0
```

```
elif msg == b"toggle":
    led.value(state)
    state = 1 - state
```

```
def main(server=SERVER):
    c = MQTTClient(CLIENT_ID, server)
    c.set_callback(sub_cb)
    c.connect()
    c.subscribe(TOPIC)
    print("Connected %s, sub to %s topic"
    % (server, TOPIC))
```

```
try:
    while 1:
        c.wait_msg()
finally:
        c.disconnect()
```