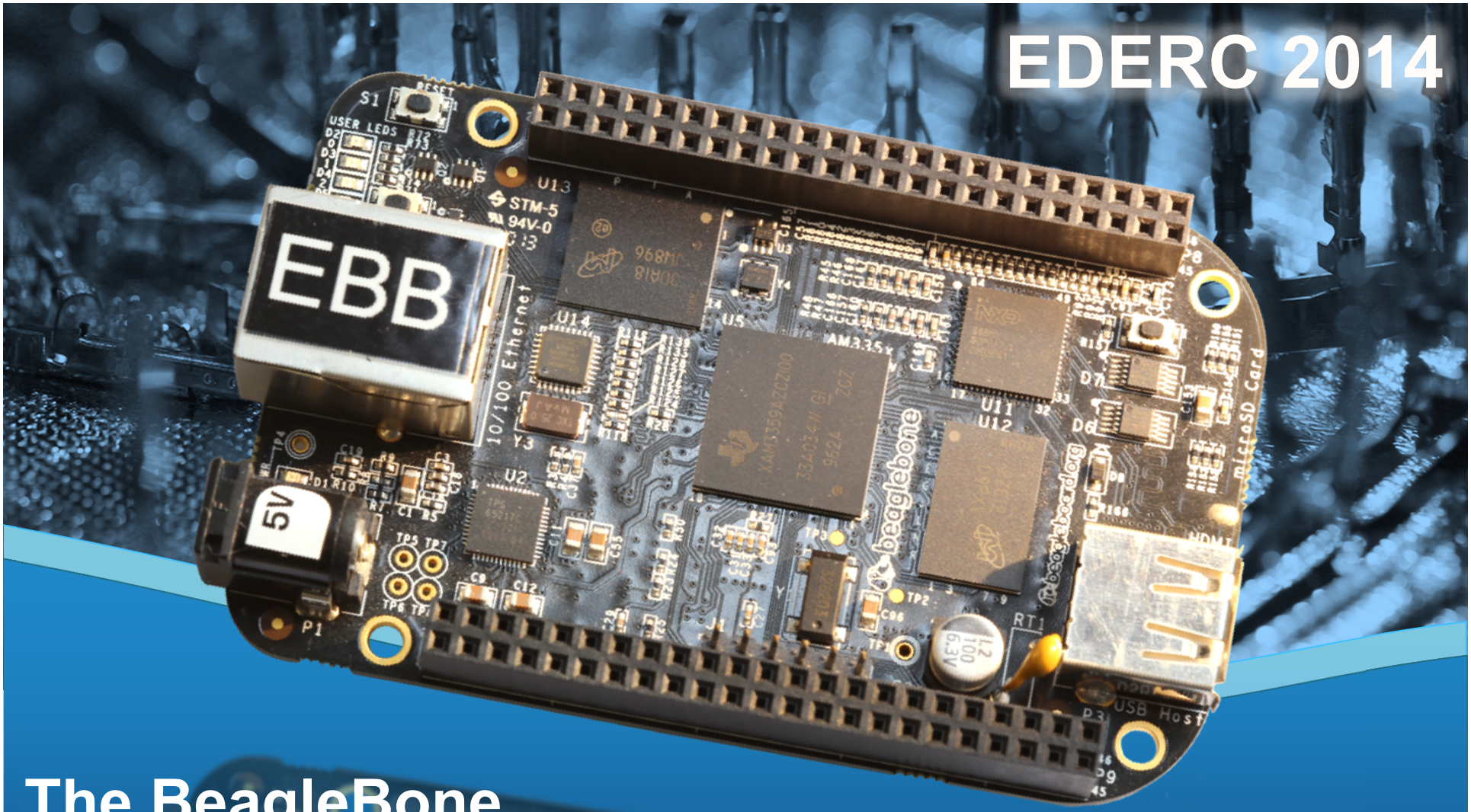
EDERC 2014



The BeagleBone
Application in Engineering Education

Dr Derek Molloy, School of Electronic Engineering,
Dublin City University, Ireland

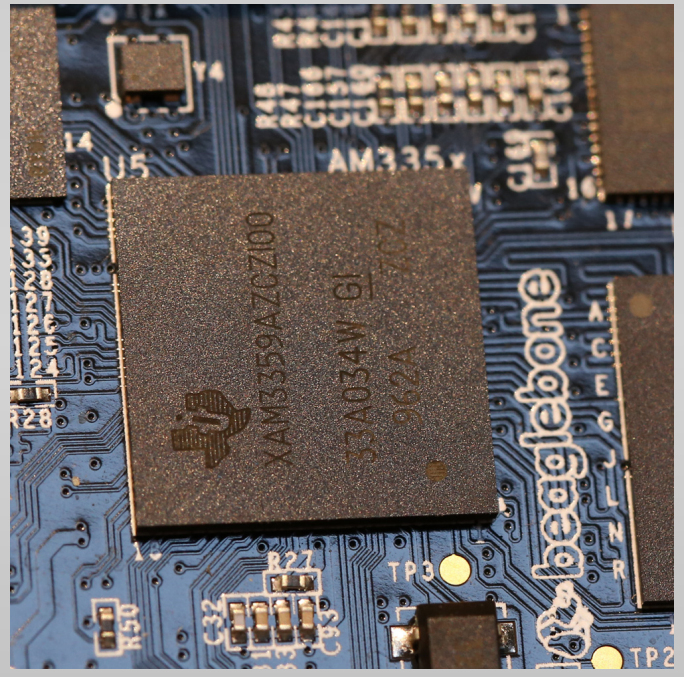
DCU

Overview

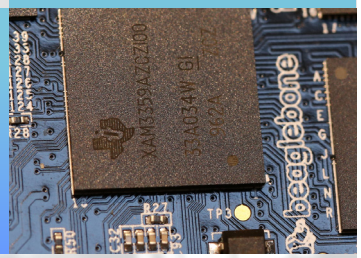
The BeagleBone: Application in Engineering Education

✓

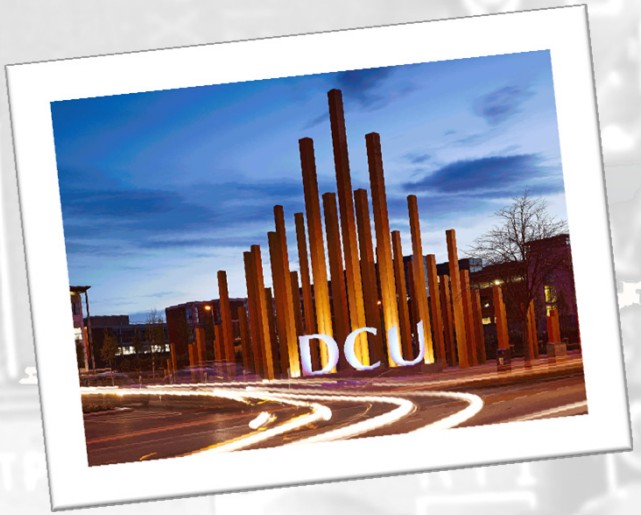
1	**Introduction**
2	BeagleBone Black Hardware
3	BeagleBone Black Software
4	Software Development Tools
5	Interfacing to the BBB
6	Applications
7	Real-Time BeagleBone
8	Conclusions



Introduction
Who am I?



- Dublin City University
  - 12,000 registered students
- Faculty of Engineering & Computing
- Research
  - Computer Vision, 3D Graphics
- Teaching
  - Electronics, 3D Graphics,
  - OOP & Embedded
- User of the Beaglebone!

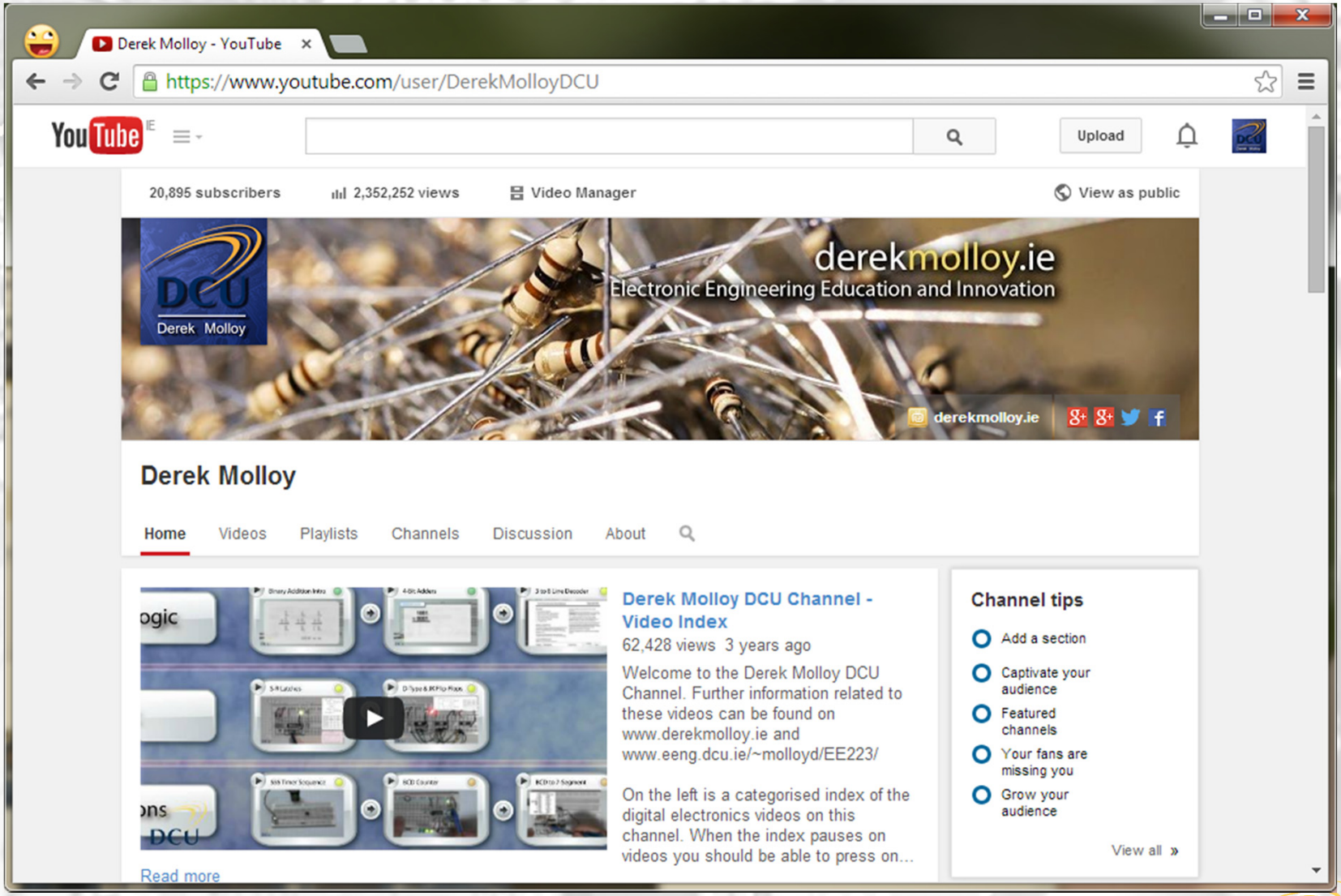


Teaching Innovations

EE223 – Digital & Analogue Electronics (5 ECTS)

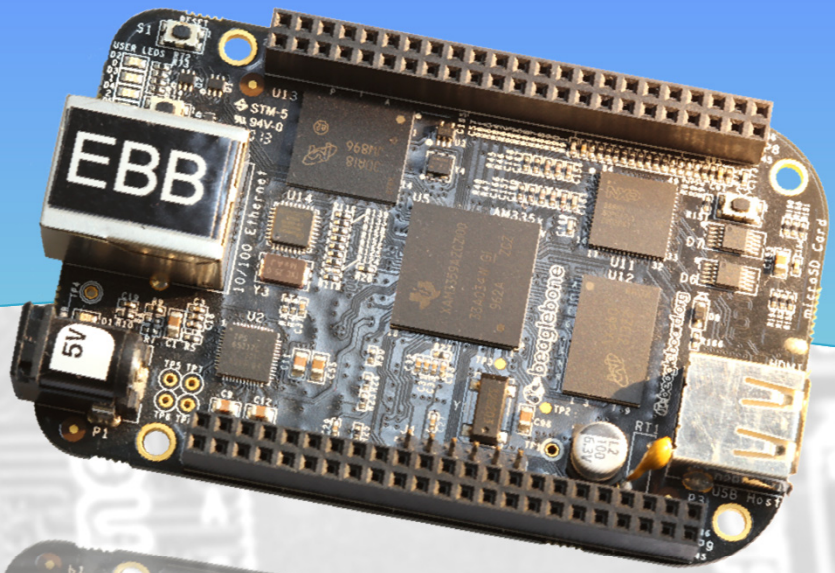
- 140 students p.a.
- Flipped labs
  - Borrow kit of components for a semester
  - Replace components free of charge (assume consumption)
  - Built in to summative assignments
- Encourage learning-by-discovery
- Supported by a YouTube channel
- Also deployed to fully on-line modality (DCU Connected)

www.youtube.com/DerekMolloyDCU



Teaching Innovations

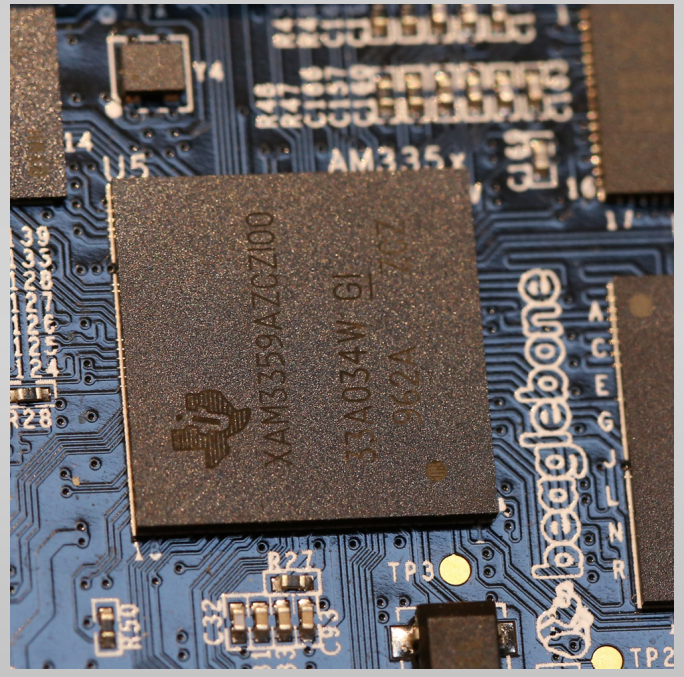
EE402 – OOP with Embedded Systems (7.5 ECTS)



- 90-100 Students p.a.
- Object-oriented Programming
- C++, Qt, Java, embedded Linux
- Assignments, computer-based exam
  - Beaglebone-based – wrap low-level hardware
  - TCP Client/Server assignment (IoT-like)
  - Supported by videos (screencast & YouTube)

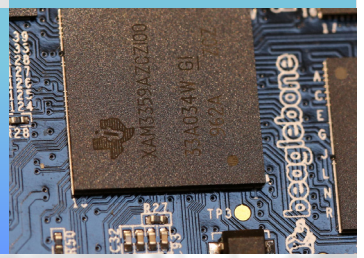
Overview

The BeagleBone: Application in Engineering Education



BeagleBone Black Hardware

Summary specification



- AM335x 1GHz ARM A8 (2,000 MIPS)
- 512 MB DDR3 RAM
- 4 GB eMMC (rev.C) (plus SD card)
- HDMI Video output (3D graphics engine)
- 10/100 Ethernet (Wi-Fi, Bluetooth via USB)
- Huge range of interfaces (GPIOs, buses, USB)
- Low power (1W to 2.3W)

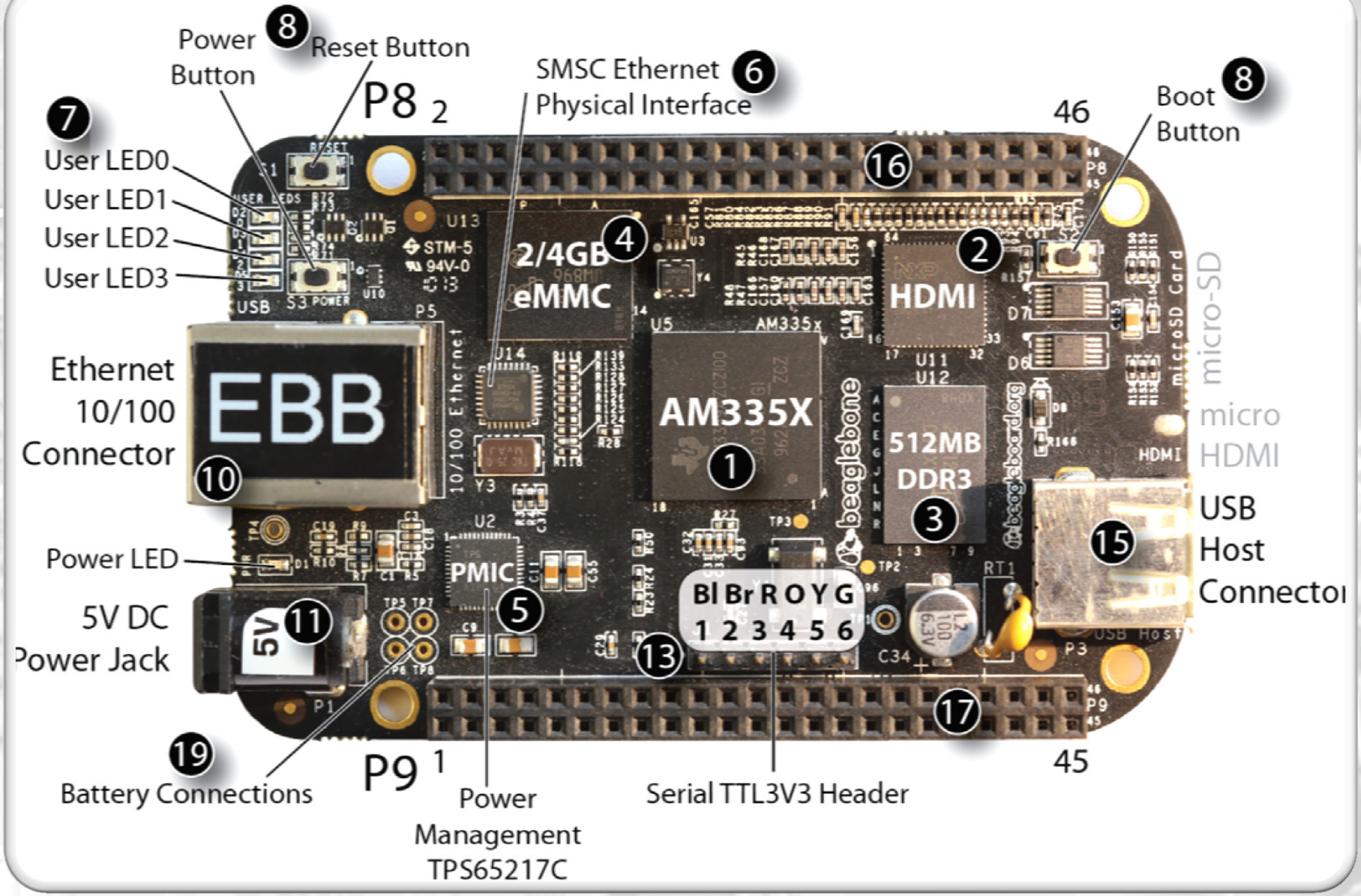
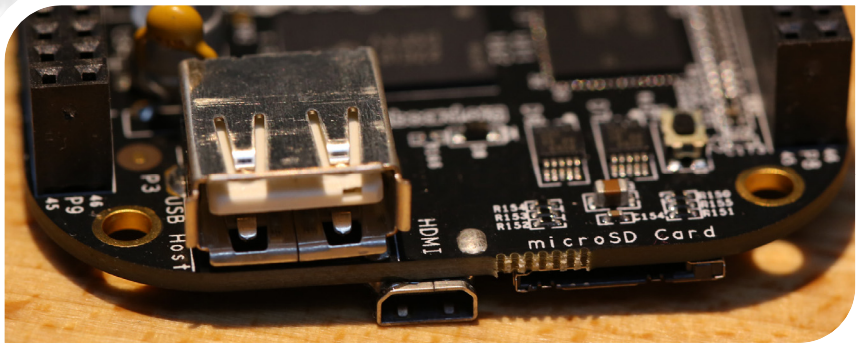
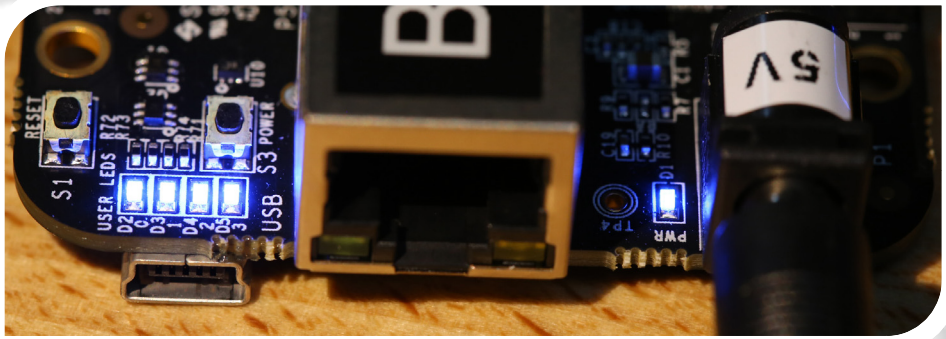
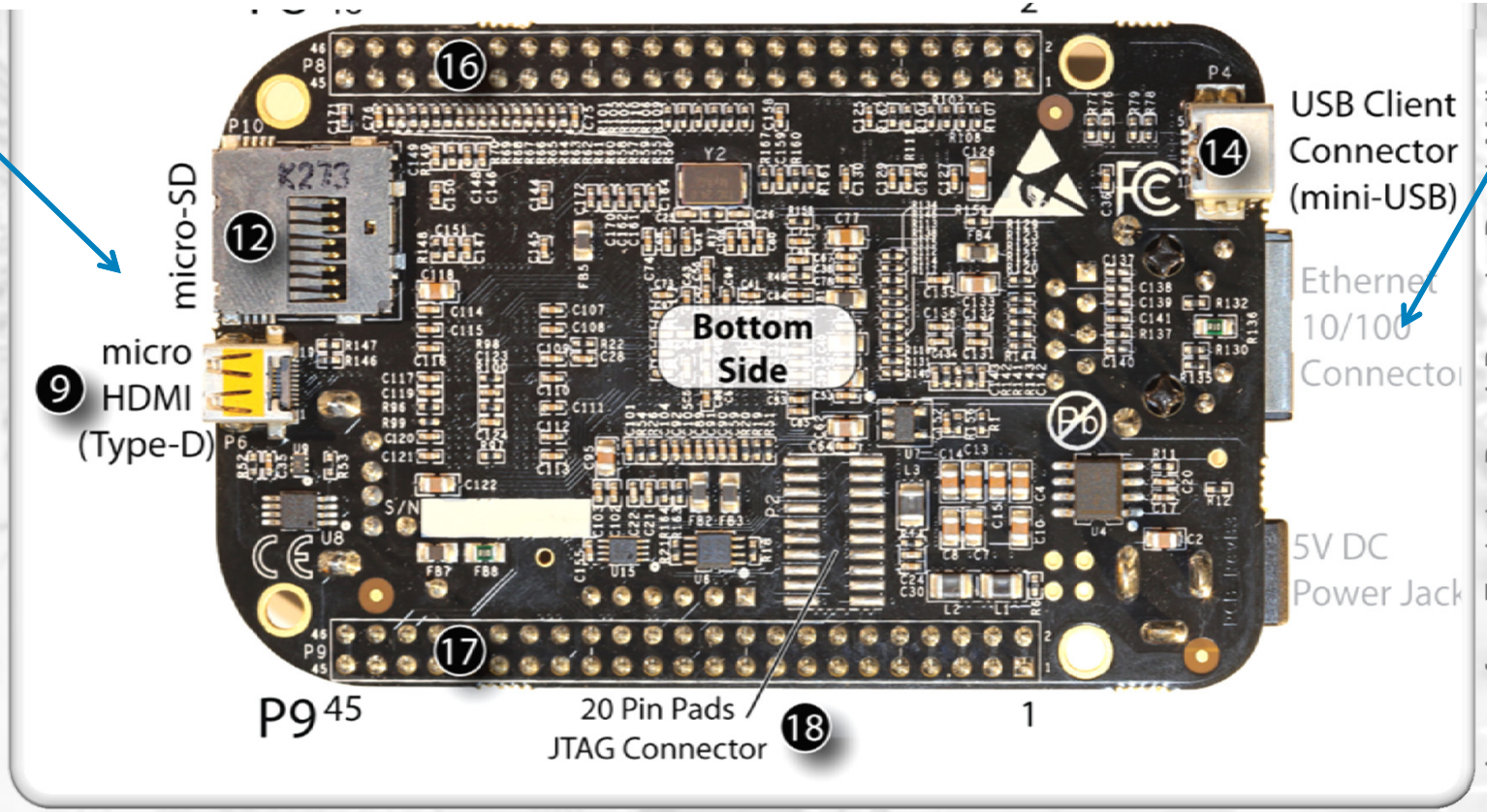


Image from Exploring BeagleBone, by Derek Molloy


USB Host
HDMI
microSD Card


RESET
USER LEDS
USB
POWER
PWR

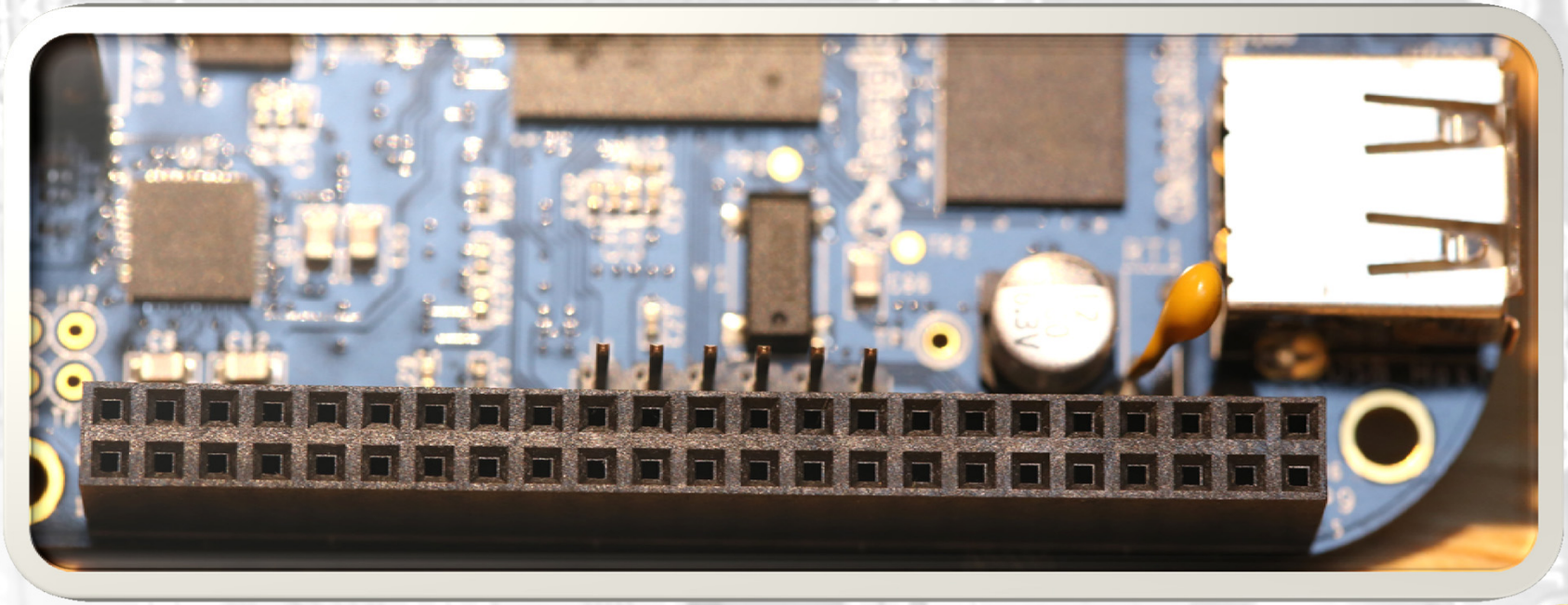

16
micro-SD
12
micro
HDMI
(Type-D)
9
Bottom
Side
USB Client
Connector
(mini-USB)
14
Ethernet
10/100
Connector
5V DC
Power Jack
17
P9 45
20 Pin Pads
JTAG Connector
18
1
Image from Exploring BeagleBone, by Derek Molloy

P8 and P9 Headers

GPIOs (x65)
PWM (x8)
Analogue Inputs (x7)
Timers (x4)
Supplies (5V, 3.3V, 1.8V)

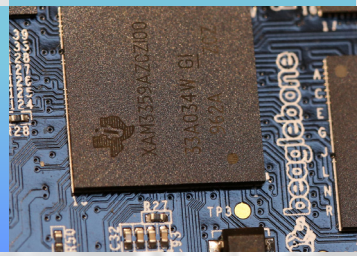
Buses

I^2C (x2)
UART (x4)
CAN Bus (x2)
SPI (x2)
GPMC
MMC (x2)
LCD
McASP (x2)

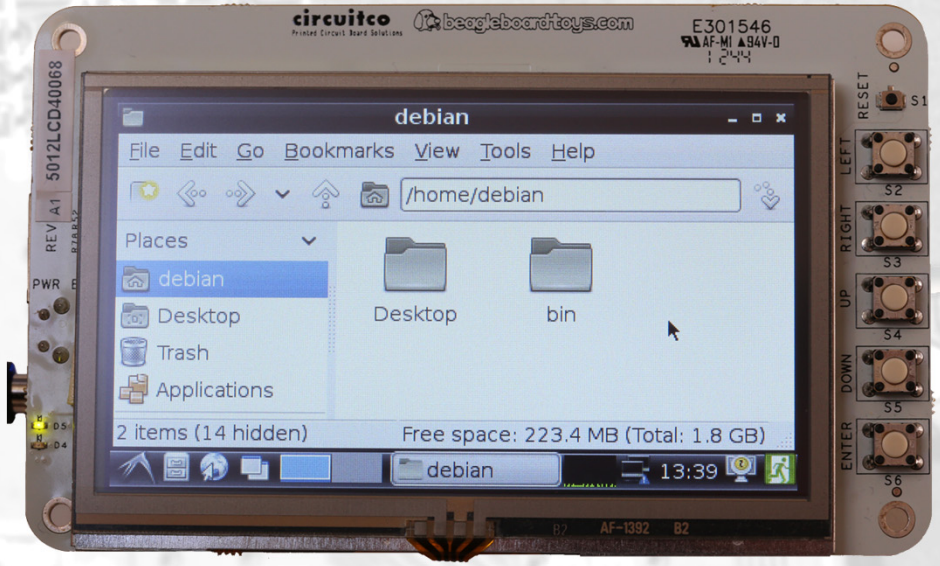


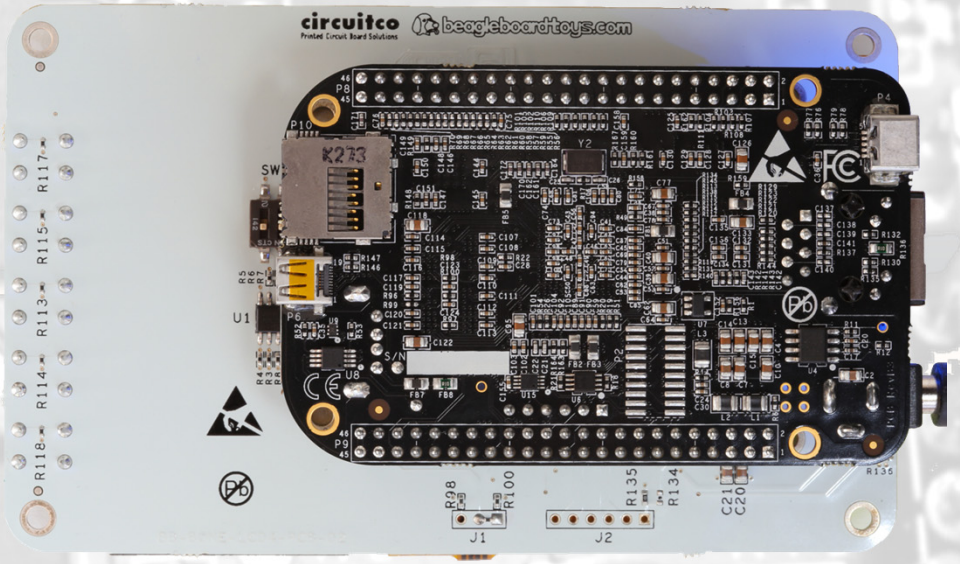
BeagleBone Black Hardware

Capes



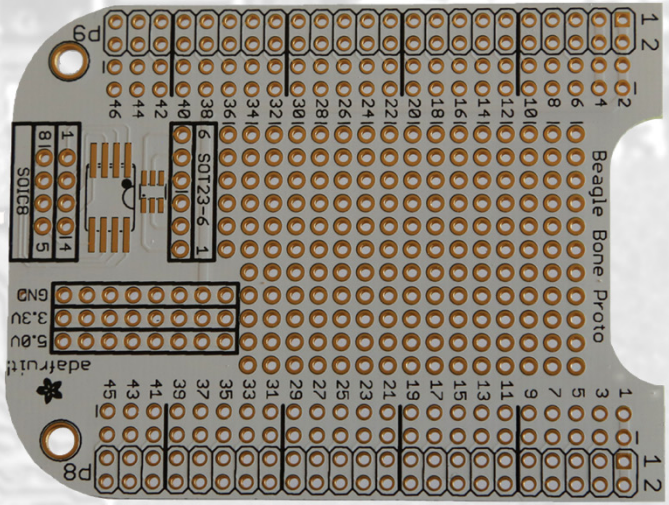
- Daughter boards
  - Attach to P8/P9 headers (stackable)



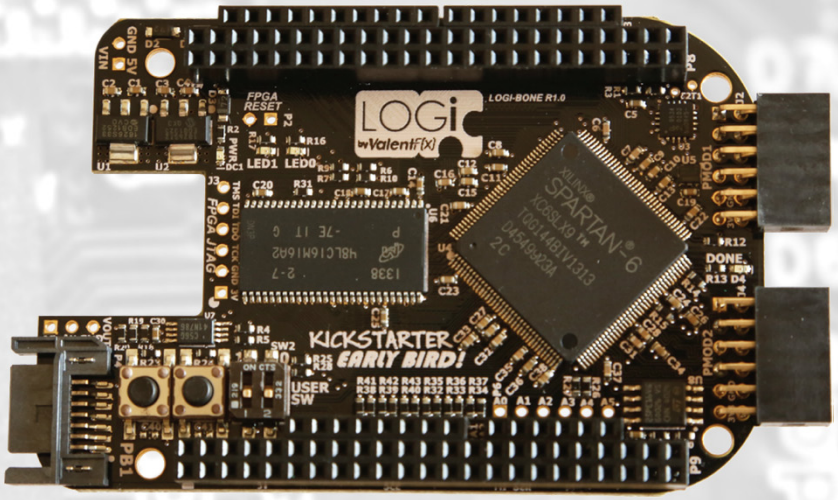


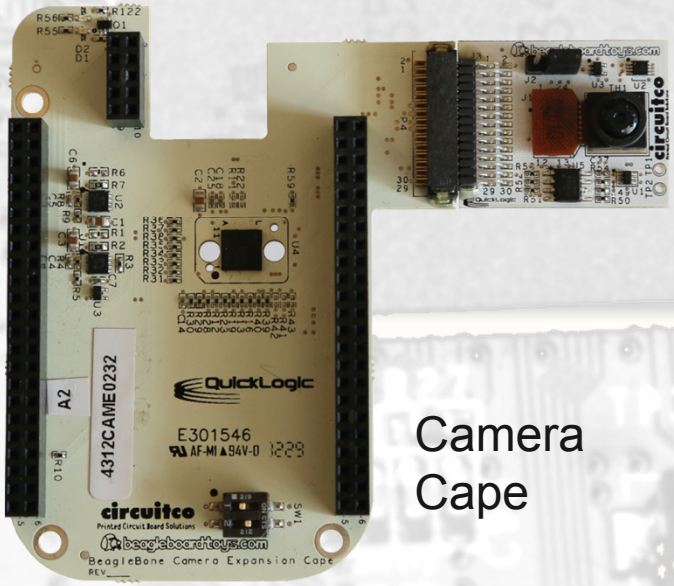
CircuitCo LCD4 Cape

Proto Cape

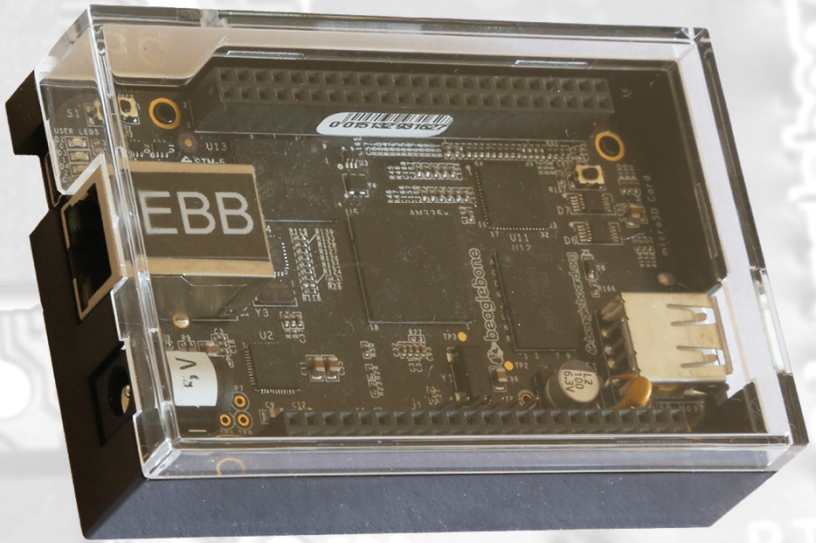


ValentF(x) FPGA Cape





Camera Cape



AdaFruit Case

Image from Exploring BeagleBone, by Derek Molloy

Peripherals

Hardware Comparison

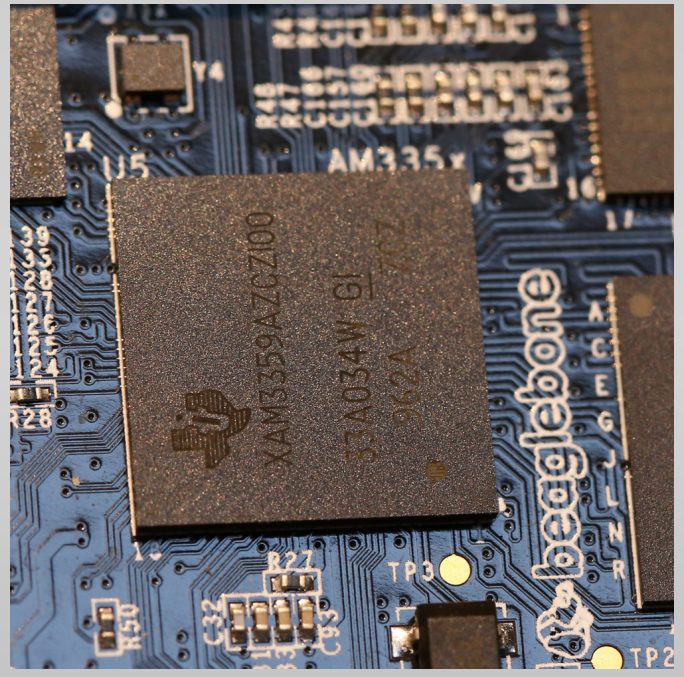
BeagleBone Black versus Raspberry PI B+, Intel Galileo

BeagleBone Black	Raspberry PI B+	Intel Galileo
1	2	3
ARM A8 1 GHz ✓	ARM A11 700 MHz	32-bit 400 MHz Quark
512MB RAM	512MB RAM	256MB RAM
$45-55	$40 ✓	$80
HDMI Video (not full HD)	HDMI Video Full HD ✓	No video
Ethernet 100	Ethernet 100	Ethernet 100
Key Features:	**Key Features:**	**Key Features:**
• eMMC 2GB/4GB ✓ • Micro SD • 2 x Programmable real-time units ✓ • 7 x ADC inputs ✓ • 86 x GPIOs, many buses ✓ • 3D Graphics Accelerator	• 4 x USB slots ✓ • H264 h/w decoder ✓ • Micro SD • Audio jack output • 40 GPIOs • Camera and DSI display connector ✓ • 3D Graphics Accelerator	• Arduino Compatible ✓ • 6 x ADC inputs • Mini-PCI Express Slot ✓ • RS-232 Serial Port • 14 x GPIOs • 8MB NOR Flash • 12-bit PWM available

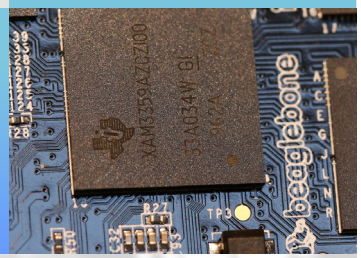
Overview

The BeagleBone: Application in Engineering Education

	1	Introduction
	2	BeagleBone Black Hardware
✓	3	**BeagleBone Black Software**
	4	Software Development Tools
	5	Interfacing to the BBB
	6	Applications
	7	Real-Time BeagleBone
	8	Conclusions



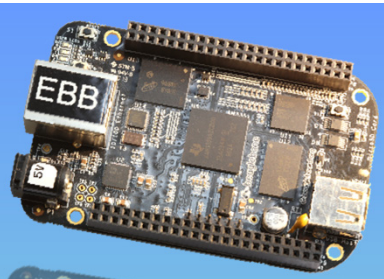
BeagleBone Software
Embedded Linux



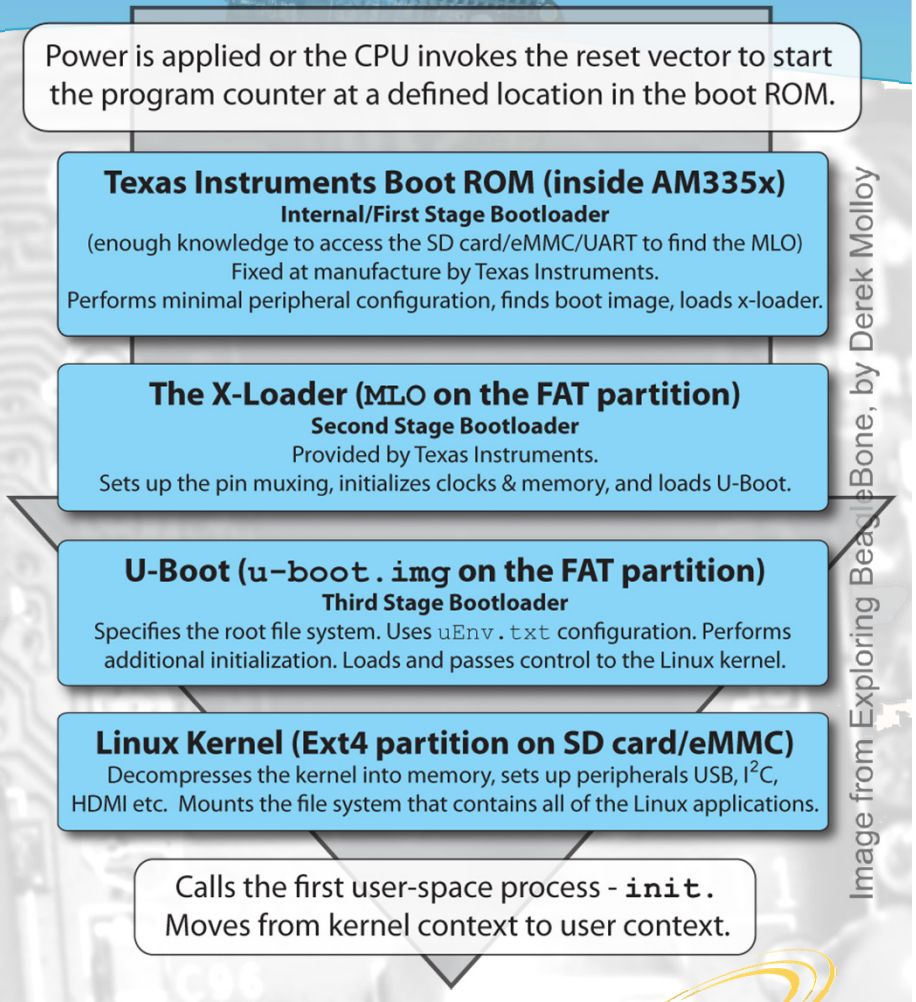
- No such thing!
  - mainline “Linux on an embedded system”
- Embedded Linux:
  - Linux is efficient and scalable
  - Huge number of open-source programs and tools
  - Excellent support for peripherals and devices
  - Downside for real-time – non-preemptive by default
- Are non-Linux solutions:
  - TI StarterWare for ARM-based Sitara Processors
  - QNX Neutrino RTOS on OMAP and Sitara

BeagleBone Black (BBB)

Linux on the BBB



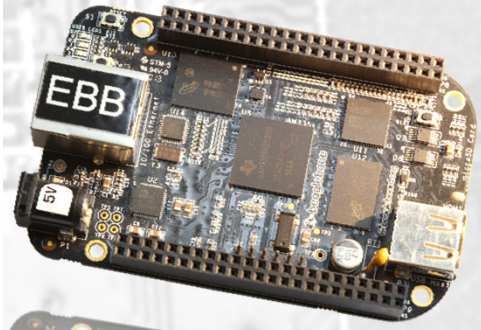
- Linux Distributions for BBB:
  - Debian – specifically packaged.
  - Ångström
  - Ubuntu, Arch etc.
- Boot from eMMC
- Boot from SD – using boot image
- Flash eMMC
  - Use flasher image from SD card
- See www.beagleboard.org

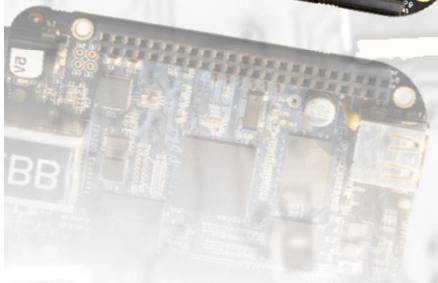


Connecting to the BeagleBone (Windows):

Connecting to the BBB

Physical Connections





Internet-over-USB

network is complex

Regular Ethernet

regular network

Cross-Over Cable

network is complex

fallback

Serial-over-USB

network failure

Serial USB-3.3V-TTL

board problems

Chrome App SSH

Web/Mac

PuTTY SSH

Windows

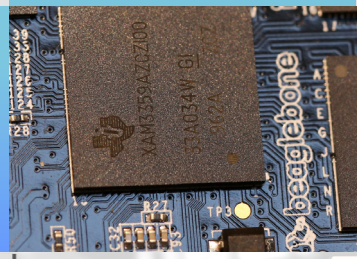
ssh/sftp

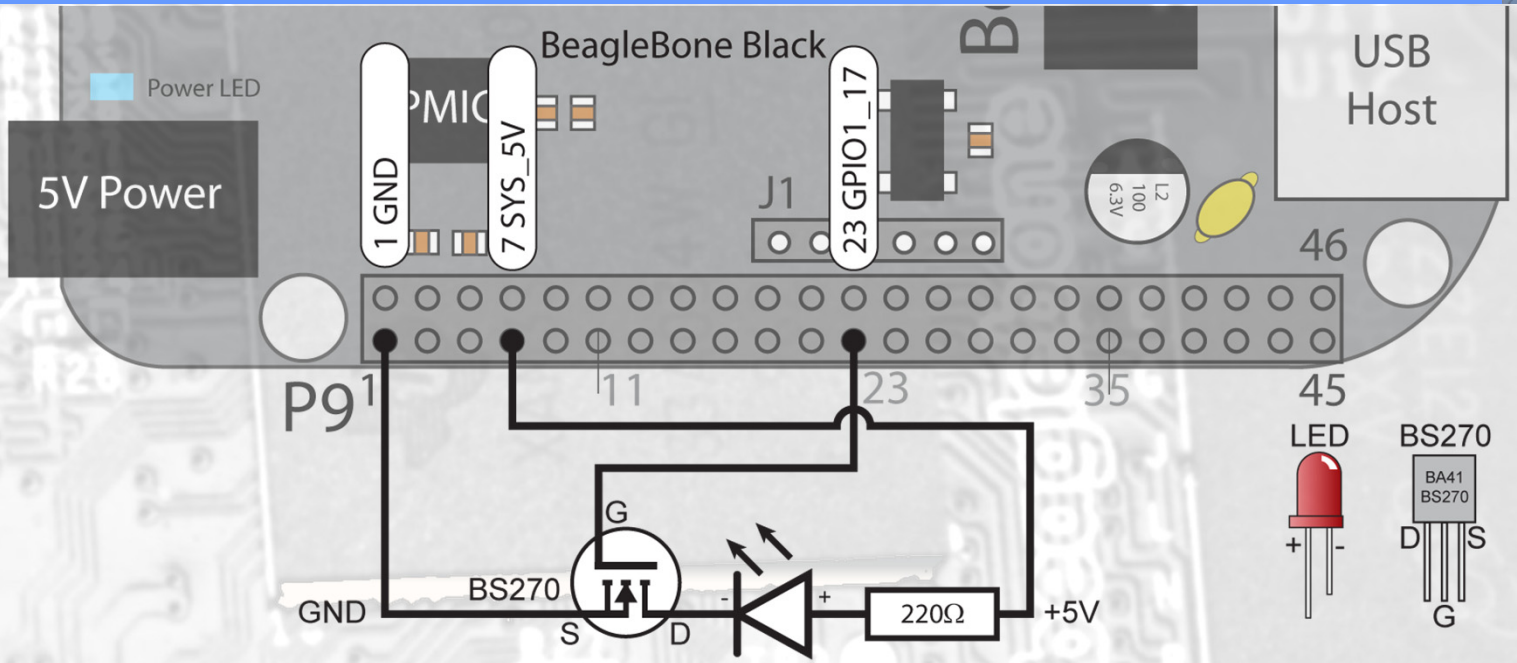
Linux

Serial Terminal

All Platforms

BeagleBone Black
A First Circuit Example



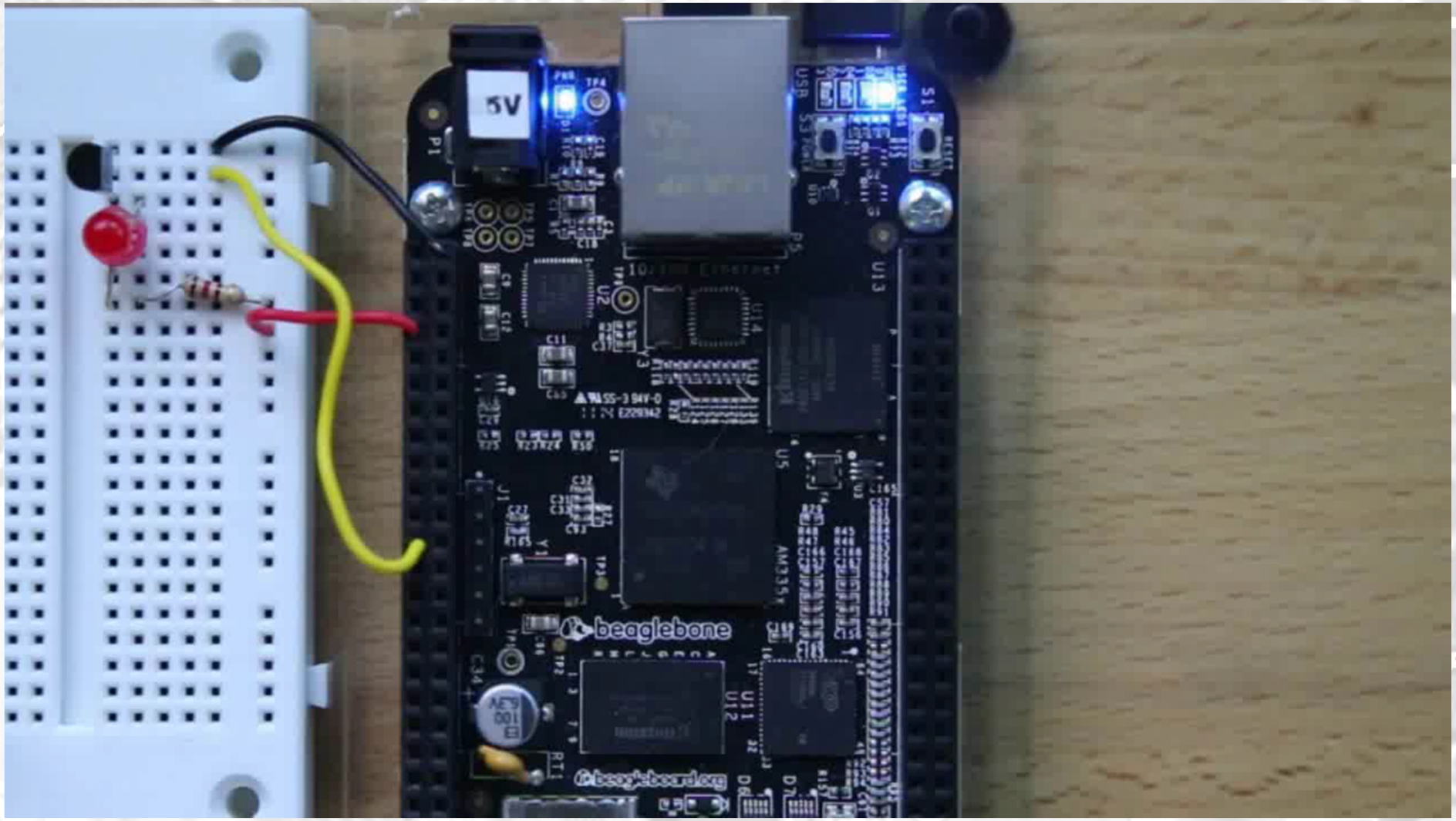


Use FET/BJT to limit GPIO to 3.3V below 4mA.

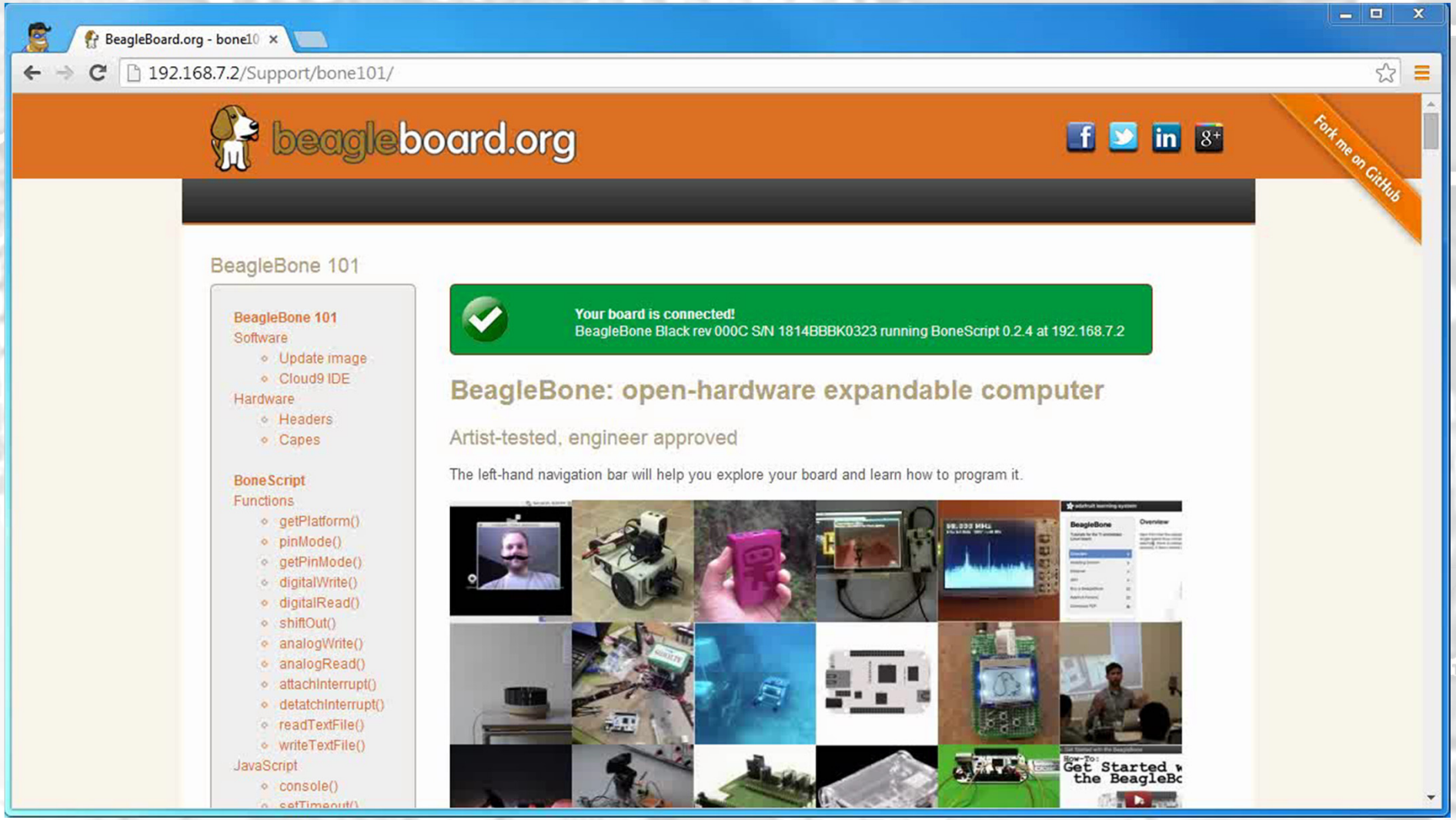
GPIO1_17 is GPIO(1x32)+17 = GPIO49

Image from Exploring BeagleBone, by Derek Molloy

Connecting to the BeagleBone (Windows):



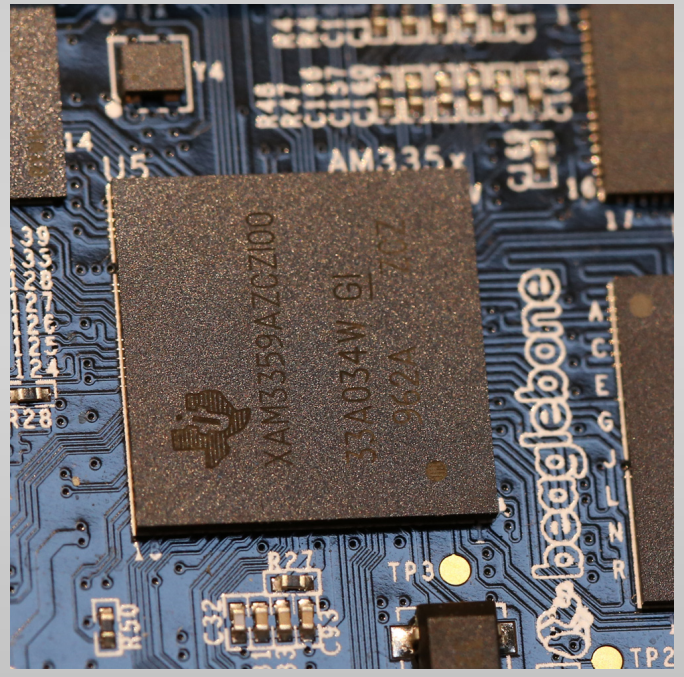
Cloud9 IDE, nodejs and BoneScript Example:



Overview

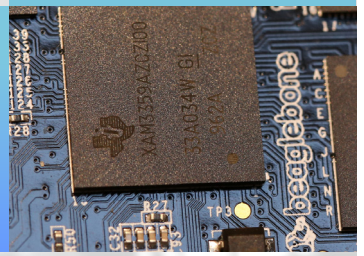
The BeagleBone: Application in Engineering Education

1. Introduction
2. BeagleBone Black Hardware
3. BeagleBone Black Software
4. ✓ **Software Development Tools**
5. Interfacing to the BBB
6. Applications
7. Real-Time BeagleBone
8. Conclusions



Software Development Tools

Building C/C++ on the BBB




```
molloyd@beaglebone: ~
molloyd@beaglebone:~$ ls *.cpp
testEDERC.cpp
molloyd@beaglebone:~$ more testEDERC.cpp
#include <iostream>
using namespace std;

int main(){
        cout << "Hello EDERC 2014!" << endl;
        return 0;
}
molloyd@beaglebone:~$ g++ testEDERC.cpp -o testEDERC
molloyd@beaglebone:~$ ./testEDERC
Hello EDERC 2014!
molloyd@beaglebone:~$
```

Software Development Tools
Cross-Platform Toolchain

- Difficult building large-scale projects on BBB
- Cross-development brings:
  - Typically faster build times
  - Single development point – multiple BBB boards
  - Rich UI development environments
- Need a Toolchain
  - Tools (e.g., gcc, gdb) and libraries (e.g., glibc)

Cross-compile test running on desktop (64-bit x86) Linux:

```
molloyd@debian:~$ sudo apt-get install g++-4.7-arm-linux-gnueabihf
…

molloyd@debian:~$ nano testToolchain.cpp
molloyd@debian:~$ more testToolchain.cpp

#include<iostream>
using namespace std;

int main(){
   cout << "Testing Toolchain" << endl;
   return 0;
}
molloyd@debian:~$ arm-linux-gnueabihf-g++ testToolchain.cpp -o testARM
```

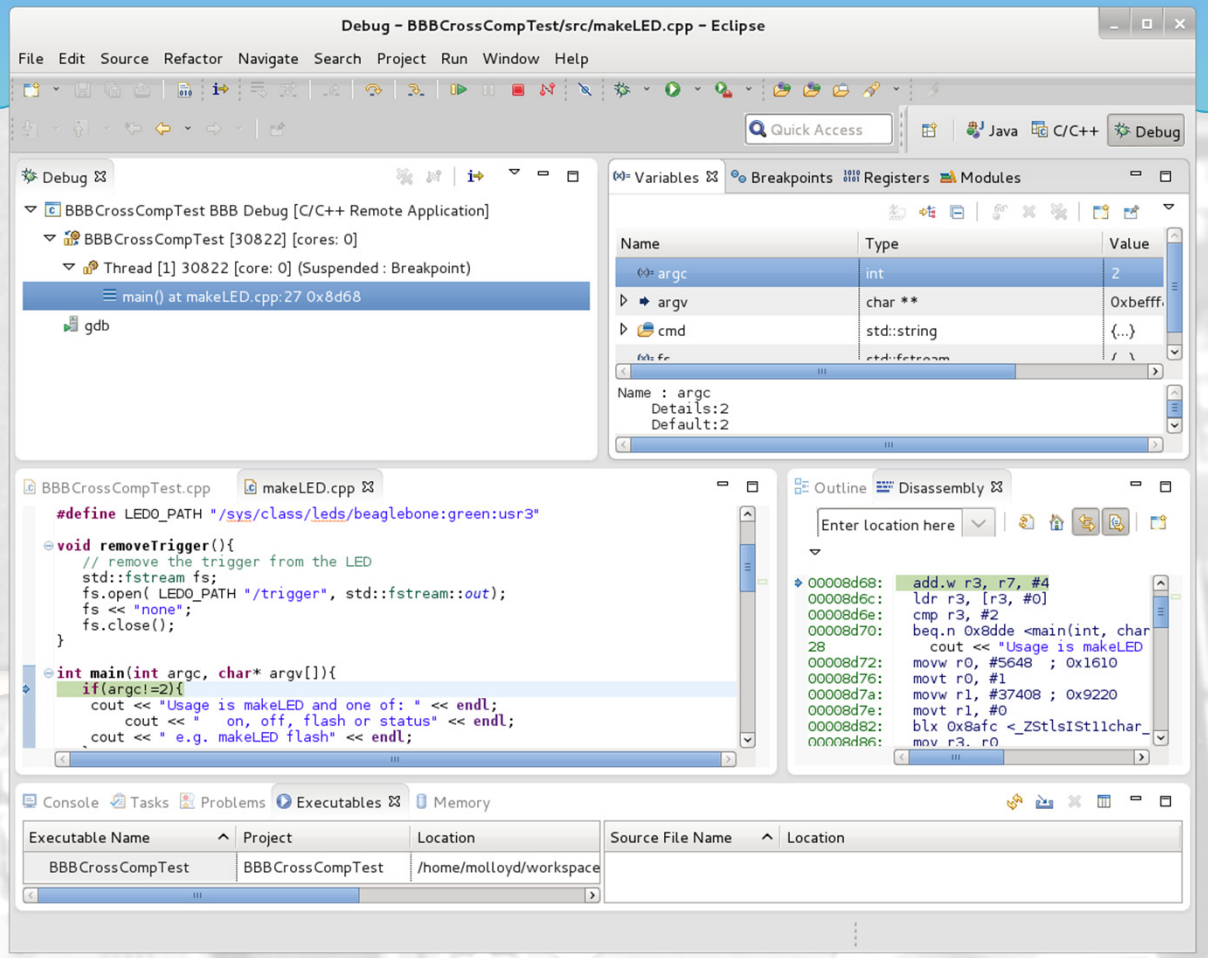
- Transfer to BeagleBone (sftp, scp, rsync…) and execute on ARMHF
- Can install a chroot and QEMU to simulate ARM on the desktop Linux image
- Better to link to Integrated Development Environment (IDE) – e.g., Eclipse, Qt Creator

Software Development Tools

Eclipse CDT

Supports:

- Cross Platform Toolchains
- Multiple Languages
- Remote System Explorer (RSE)
- Remote Debug
- Git/GitHub Integration
- Doxygen Integration

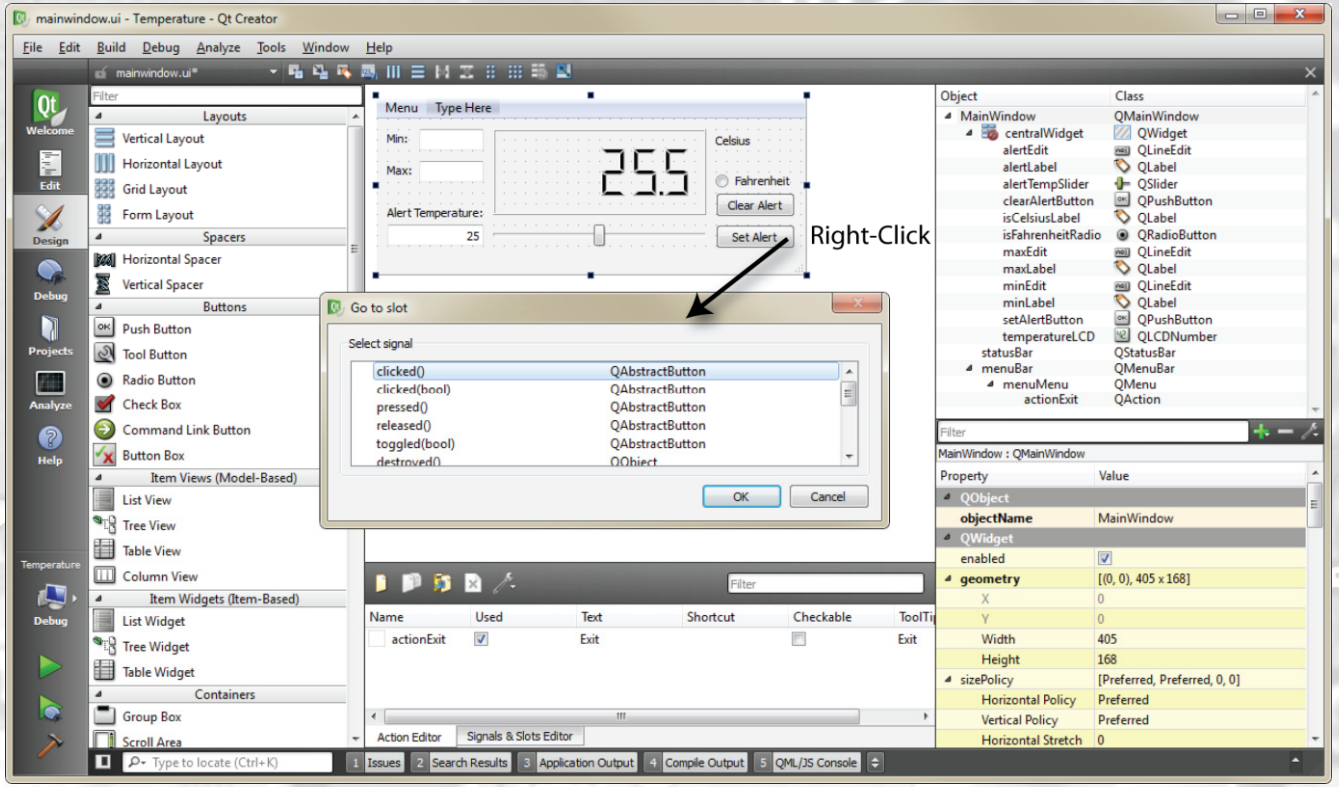


Software Development Tools

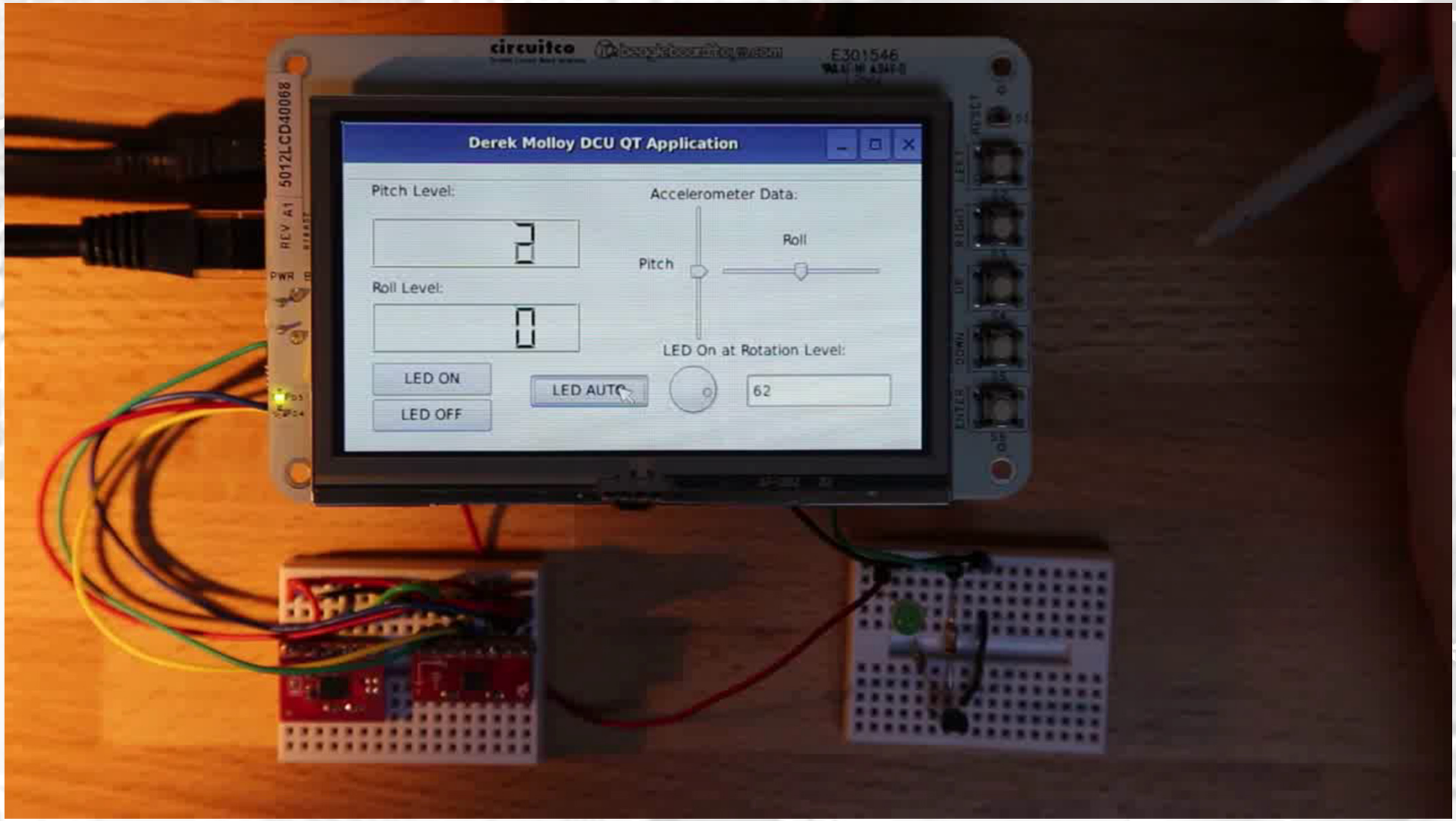
Qt Creator

Supports:

- Cross Platform Toolchains
- Qt GUI Tools
- Remote System Support
- Remote Deploy & Debug Support
- Sockets, Threads, Networking etc.



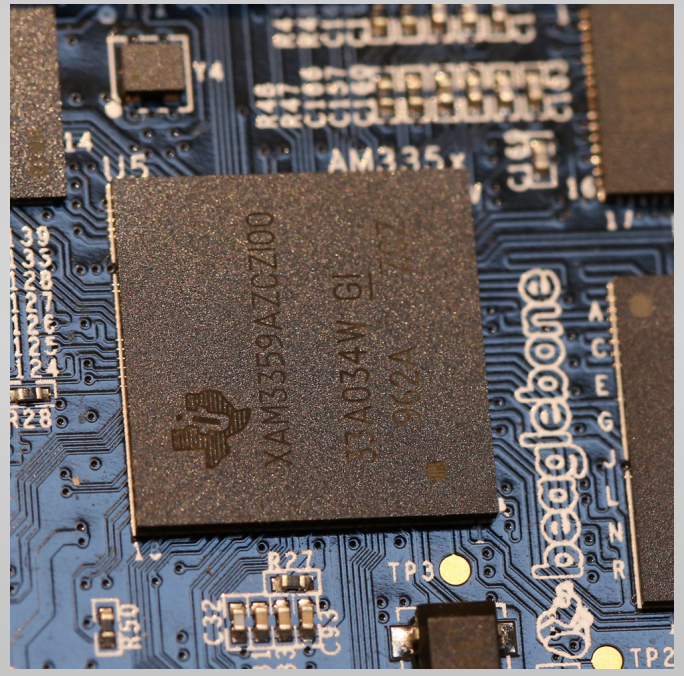
Example Qt Integration Project (display, sensors, UI …)



Overview

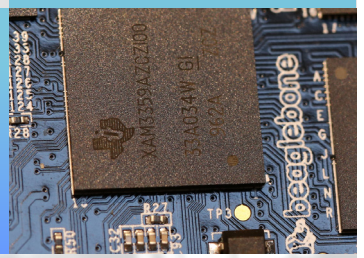
The BeagleBone: Application in Engineering Education

1. Introduction
2. BeagleBone Black Hardware
3. BeagleBone Black Software
4. Software Development Tools
5. ✓ **Interfacing to the BBB**
6. Applications
7. Real-Time BeagleBone
8. Conclusions



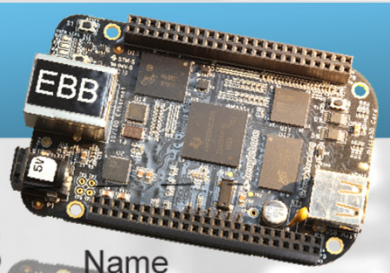
Interfacing to the BBB

Common Interface Types



- GPIO Digital Output/Input
- Analog Input
- PWM Output
- Bus interfaces (e.g., I^2C, SPI, UART)
- USB Devices
- The AM3358/9 has BGA with 324 pins
  - Only 2 x 46 pin headers on the BBB!
  - Need pin mux

Default P8 and P9 Pin assignments



Name	P9		Name
GND	P9_01	P9_02	GND
DC_3.3V	P9_03	P9_04	DC_3.3V
VDD_5V	P9_05	P9_06	VDD_5V
SYS_5V	P9_07	P9_08	SYS_5V
PWR_BUT	P9_09	P9_10	SYS_RESETn
UART4_RXD	P9_11	P9_12	GPIO1_28
UART4_TXD	P9_13	P9_14	EHRPWM1A
GPIO1_16	P9_15	P9_16	EHRPWM1B
I2C1_SCL	P9_17	P9_18	I2C1_SDA
I2C2_SCL	P9_19	P9_20	I2C2_SDA
UART2_TXD	P9_21	P9_22	UART2_RXD
GPIO1_17	P9_23	P9_24	UART1_TXD
GPIO3_21	P9_25	P9_26	UART1_RXD
GPIO3_19	P9_27	P9_28	SPI1_CS0
SPI1_D0	P9_29	P9_30	SPI1_D1
SPI1_SCLK	P9_31	P9_32	VADC
AIN4	P9_33	P9_34	AGND
AIN6	P9_35	P9_36	AIN5
AIN2	P9_37	P9_38	AIN3
AIN0	P9_39	P9_40	AIN1
GPIO3_20	P9_41	P9_42	GPIO0_7
GND	P9_43	P9_44	GND
GND	P9_45	P9_46	GND

Name	P8		Name
GND	P8_01	P8_02	GND
GPIO1_6	P8_03	P8_04	GPIO1_7
GPIO1_2	P8_05	P8_06	GPIO1_3
TIMER4	P8_07	P8_08	TIMER7
TIMER5	P8_09	P8_10	TIMER6
GPIO1_13	P8_11	P8_12	GPIO1_12
EHRPWM2B	P8_13	P8_14	GPIO0_26
GPIO1_15	P8_15	P8_16	GPIO1_14
GPIO0_27	P8_17	P8_18	GPIO2_1
EHRPWM2A	P8_19	P8_20	GPIO1_31
GPIO1_30	P8_21	P8_22	GPIO1_5
GPIO1_4	P8_23	P8_24	GPIO1_1
GPIO1_0	P8_25	P8_26	GPIO1_29
GPIO2_22	P8_27	P8_28	GPIO2_24
GPIO2_23	P8_29	P8_30	GPIO2_25
UART5_CTSN	P8_31	P8_32	UART5_RTSN
UART4_RTSN	P8_33	P8_34	UART3_RTSN
UART4_CTSN	P8_35	P8_36	UART3_CTSN
UART5_TXD	P8_37	P8_38	UART5_RXD
GPIO2_12	P8_39	P8_40	GPIO2_13
GPIO2_10	P8_41	P8_42	GPIO2_11
GPIO2_8	P8_43	P8_44	GPIO2_9
GPIO2_6	P8_45	P8_46	GPIO2_7

USB

Image from Exploring BeagleBone, by Derek Molloy

P8 Header Pin Allocations:

Pin	$PINS	ADDR	GPIO	Name	Mode7	Mode6	Mode5	Mode4	Mode3	Mode2	Mode1	Mode0	CPU	Notes
P8_01		Offset from:		DGND										Ground
P8_02		44e10800		DGND										Ground
P8_03	6	0x818/018	38	GPIO1_6	gpio1[6]						mmc1_dat6	gpmc_ad6	R9	Allocated emmc2
P8_04	7	0x81c/01c	39	GPIO1_7	gpio1[7]						mmc1_dat7	gpmc_ad7	T9	Allocated emmc2
P8_05	2	0x808/008	34	GPIO1_2	gpio1[2]						mmc1_dat2	gpmc_ad2	R8	Allocated emmc2
P8_06	3	0x80c/00c	35	GPIO1_3	gpio1[3]						mmc1_dat3	gpmc_ad3	T8	Allocated emmc2
P8_07	36	0x890/090	66	TIMER4	gpio2[2]					timer4		gpmc_advn_ale	R7	
P8_08	37	0x894/094	67	TIMER7	gpio2[3]					timer7		gpmc_oen_ren	T7	
P8_09	39	0x89c/09c	69	TIMER5	gpio2[5]					timer5		gpmc_be0n_cle	T6	
P8_10	38	0x898/098	68	TIMER6	gpio2[4]					timer6		gpmc_wen	U6	
P8_11	13	0x834/034	45	GPIO1_13	gpio1[13]	pr1_pru0_pru_r30_15		eQEP2B_in	mmc2_dat1	mmc1_dat5	lcd_data18	gpmc_ad13	R12	
P8_12	12	0x830/030	44	GPIO1_12	gpio1[12]	pr1_pru0_pru_r30_14		EQEP2A_IN	MMC2_DAT0	MMC1_DAT4	LCD_DATA19	GPMC_AD12	T12	
P8_13	9	0x824/024	23	EHRPWM2B	gpio0[23]			ehrpwm2B	mmc2_dat5	mmc1_dat1	lcd_data22	gpmc_ad9	T10	
P8_14	10	0x828/028	26	GPIO0_26	gpio0[26]			ehrpwm2_tripzone_in	mmc2_dat6	mmc1_dat2	lcd_data21	gpmc_ad10	T11	
P8_15	15	0x83c/03c	47	GPIO1_15	gpio1[15]	pr1_pru0_pru_r31_15		eQEP2_strobe	mmc2_dat3	mmc1_dat7	lcd_data16	gpmc_ad15	U13	
P8_16	14	0x838/038	46	GPIO1_14	gpio1[14]	pr1_pru0_pru_r31_14		eQEP2_index	mmc2_dat2	mmc1_dat6	lcd_data17	gpmc_ad14	V13	
P8_17	11	0x82c/02c	27	GPIO0_27	gpio0[27]			ehrpwm0_synco	mmc2_dat7	mmc1_dat3	lcd_data20	gpmc_ad11	U12	
P8_18	35	0x88c/08c	65	GPIO2_1	gpio2[1]	mcasp0_fsr			mmc2_clk	gpmc_wait1	lcd_memory_clk	gpmc_clk_mux0	V12	
P8_19	8	0x820/020	22	EHRPWM2A	gpio0[22]			ehrpwm2A	mmc2_dat4	mmc1_dat0	lcd_data23	gpmc_ad8	U10	
P8_20	33	0x884/084	63	GPIO1_31	gpio1[31]	pr1_pru1_pru_r31_13	pr1_pru1_pru_r30_13			mmc1_cmd	gpmc_be1n	gpmc_csn2	V9	Allocated emmc2
P8_21	32	0x880/080	62	GPIO1_30	gpio1[30]	pr1_pru1_pru_r31_12	pr1_pru1_pru_r30_12			mmc1_clk	gpmc_clk	gpmc_csn1	U9	Allocated emmc2
P8_22	5	0x814/014	37	GPIO1_5	gpio1[5]						mmc1_dat5	gpmc_ad5	V8	Allocated emmc2
P8_23	4	0x810/010	36	GPIO1_4	gpio1[4]						mmc1_dat4	gpmc_ad4	U8	Allocated emmc2
P8_24	1	0x804/004	33	GPIO1_1	gpio1[1]						mmc1_dat1	gpmc_ad1	V7	Allocated emmc2
P8_25	0	0x800/000	32	GPIO1_0	gpio1[0]						mmc1_dat0	gpmc_ad0	U7	Allocated emmc2
P8_26	31	0x87c/07c	61	GPIO1_29	gpio1[29]							gpmc_csn0	V6	
P8_27	56	0x8e0/0e0	86	GPIO2_22	gpio2[22]	pr1_pru1_pru_r31_8	pr1_pru1_pru_r30_8				gpmc_a8	lcd_vsync	U5	Allocated HDMI
P8_28	58	0x8e8/0e8	88	GPIO2_24	gpio2[24]	pr1_pru1_pru_r31_10	pr1_pru1_pru_r30_10				gpmc_a10	lcd_pclk	V5	Allocated HDMI
P8_29	57	0x8e4/0e4	87	GPIO2_23	gpio2[23]	pr1_pru1_pru_r31_9	pr1_pru1_pru_r30_9				gpmc_a9	lcd_hsync	R5	Allocated HDMI
P8_30	59	0x8ec/0ec	89	GPIO2_25	gpio2[25]						gpmc_a11	lcd_ac_bias_en	R6	Allocated HDMI
P8_31	54	0x8d8/0d8	10	UART5_CTSN	gpio0[10]	uart5_ctsn		uart5_rxd	mcasp0_axr1	eQEP1_index	gpmc_a18	lcd_data14	V4	Allocated HDMI
P8_32	55	0x8dc/0dc	11	UART5_RTSN	gpio0[11]	uart5_rtsn		mcasp0_axr3	mcasp0_ahclkx	eQEP1_strobe	gpmc_a19	lcd_data15	T5	Allocated HDMI
P8_33	53	0x8d4/0d4	9	UART4_RTSN	gpio0[9]	uart4_rtsn		mcasp0_axr3	mcasp0_fsr	eQEP1B_in	gpmc_a17	lcd_data13	V3	Allocated HDMI
P8_34	51	0x8cc/0cc	81	UART3_RTSN	gpio2[17]	uart3_rtsn		mcasp0_axr2	mcasp0_ahclkr	ehrpwm1B	gpmc_a15	lcd_data11	U4	Allocated HDMI
P8_35	52	0x8d0/0d0	8	UART4_CTSN	gpio0[8]	uart4_ctsn		mcasp0_axr2	mcasp0_aclkr	eQEP1A_in	gpmc_a16	lcd_data12	V2	Allocated HDMI
P8_36	50	0x8c8/0c8	80	UART3_CTSN	gpio2[16]	uart3_ctsn			mcasp0_axr0	ehrpwm1A	gpmc_a14	lcd_data10	U3	Allocated HDMI
P8_37	48	0x8c0/0c0	78	UART5_TXD	gpio2[14]	uart2_ctsn		uart5_txd	mcasp0_aclkx	ehrpwm1_tripzone_in	gpmc_a12	lcd_data8	U1	Allocated HDMI
P8_38	49	0x8c4/0c4	79	UART5_RXD	gpio2[15]	uart2_rtsn		uart5_rxd	mcasp0_fsx	ehrpwm0_synco	gpmc_a13	lcd_data9	U2	Allocated HDMI
P8_39	46	0x8b8/0b8	76	GPIO2_12	gpio2[12]	pr1_pru1_pru_r31_6	pr1_pru1_pru_r30_6		eQEP2_index		gpmc_a6	lcd_data6	T3	Allocated HDMI
P8_40	47	0x8bc/0bc	77	GPIO2_13	gpio2[13]	pr1_pru1_pru_r31_7	pr1_pru1_pru_r30_7	pr1_edio_data_out7	eQEP2_strobe		gpmc_a7	lcd_data7	T4	Allocated HDMI
P8_41	44	0x8b0/0b0	74	GPIO2_10	gpio2[10]	pr1_pru1_pru_r31_4	pr1_pru1_pru_r30_4		eQEP2A_in		gpmc_a4	lcd_data4	T1	Allocated HDMI
P8_42	45	0x8b4/0b4	75	GPIO2_11	gpio2[11]	pr1_pru1_pru_r31_5	pr1_pru1_pru_r30_5		eQEP2B_in		gpmc_a5	lcd_data5	T2	Allocated HDMI
P8_43	42	0x8a8/0a8	72	GPIO2_8	gpio2[8]	pr1_pru1_pru_r31_2	pr1_pru1_pru_r30_2		ehrpwm2_tripzone_in		gpmc_a2	lcd_data2	R3	Allocated HDMI
P8_44	43	0x8ac/0ac	73	GPIO2_9	gpio2[9]	pr1_pru1_pru_r31_3	pr1_pru1_pru_r30_3		ehrpwm0_synco		gpmc_a3	lcd_data3	R4	Allocated HDMI
P8_45	40	0x8a0/0a0	70	GPIO2_6	gpio2[6]	pr1_pru1_pru_r31_0	pr1_pru1_pru_r30_0		ehrpwm2A		gpmc_a0	lcd_data0	R1	Allocated HDMI
P8_46	41	0x8a4/0a4	71	GPIO2_7	gpio2[7]	pr1_pru1_pru_r31_1	pr1_pru1_pru_r30_1		ehrpwm2B		gpmc_a1	lcd_data1	R2	Allocated HDMI
P9 Header	cat $PINS	ADDR +	GPIO NO.	Name	Mode 7	Mode 6	Mode 5	Mode 4	Mode 3	Mode 2	Mode 1	Mode 0	CPU	

Image from Exploring BeagleBone, by Derek Molloy

Interfacing to the BBB
Device Tree Overlays

```
…
compatible = "ti,beaglebone", "ti,beaglebone-black";
part-number = "EBB-GPIO-Example";
version = "00A0";
fragment@0 {
   target = <&am33xx_pinmux>;
   __overlay__ {
      ebb_example: EBB_GPIO_Example {
         pinctrl-single,pins = <
            0x070 0x07   // P9_11 $28 GPIO0_30=30 Output Mode7 pulldown
            0x074 0x37   // P9_13 $29 GPIO0_31=31 Input Mode7 pullup
         >;
      };
   }; …
```

- 0x27 (0100111) Fast, Input, Pull-Down, Enabled and Mux Mode 7
- 0x37 (0110111) Fast, Input, Pull-Up, Enabled, Mux Mode 7
- 0x07 (0000111) Fast, Output, Pull-down, Enabled, Mux Mode 7
- 0x17 (0010111) Fast, Output, Pull-up, Enabled, Mux Mode 7

Interfacing to the BBB

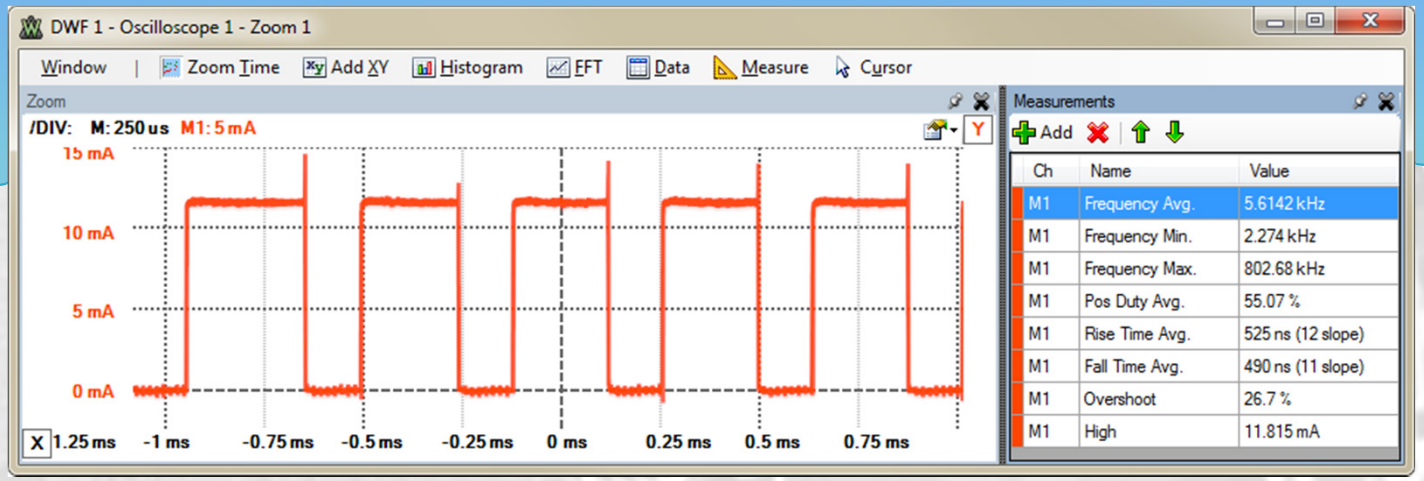
Cape Manager

```
molloyd@beaglebone:/lib/firmware$ sudo su
root@beaglebone:/lib/firmware# echo EBB-GPIO-Example > $SLOTS
root@beaglebone:/lib/firmware# cat $SLOTS
 0: 54:PF---
 1: 55:PF---
 2: 56:PF---
 3: 57:PF---
 4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,BB-BONE-EMMC-2G
 5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,BB-BONELT-HDMI
 6: ff:P-O-L Override Board Name,00A0,Override Manuf,EBB-GPIO-Example
```

- Allows pins to be allocated for capes
- Virtual capes
  - Build overlays using the device tree compiler
  - Can add and remove dynamically or on boot

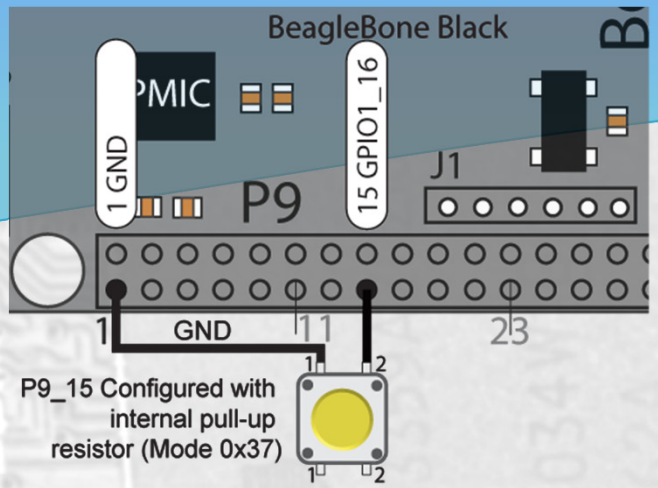
Interfacing to the BBB

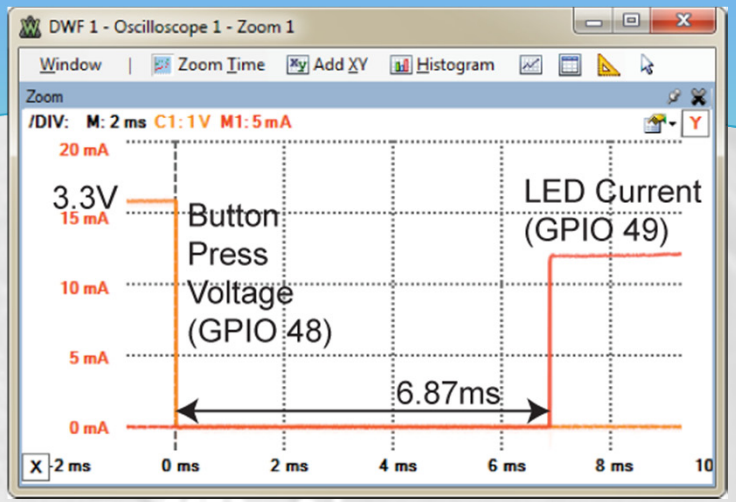
Digital Output



- C/C++ Linux userspace code available
  - Perfect for low-frequency switching
  - Limited switching frequency, suffers from jitter
  - Can directly memory switch (dangerous?)
  - Can use the PRU-ICSS

Interfacing to the BBB
Digital Input

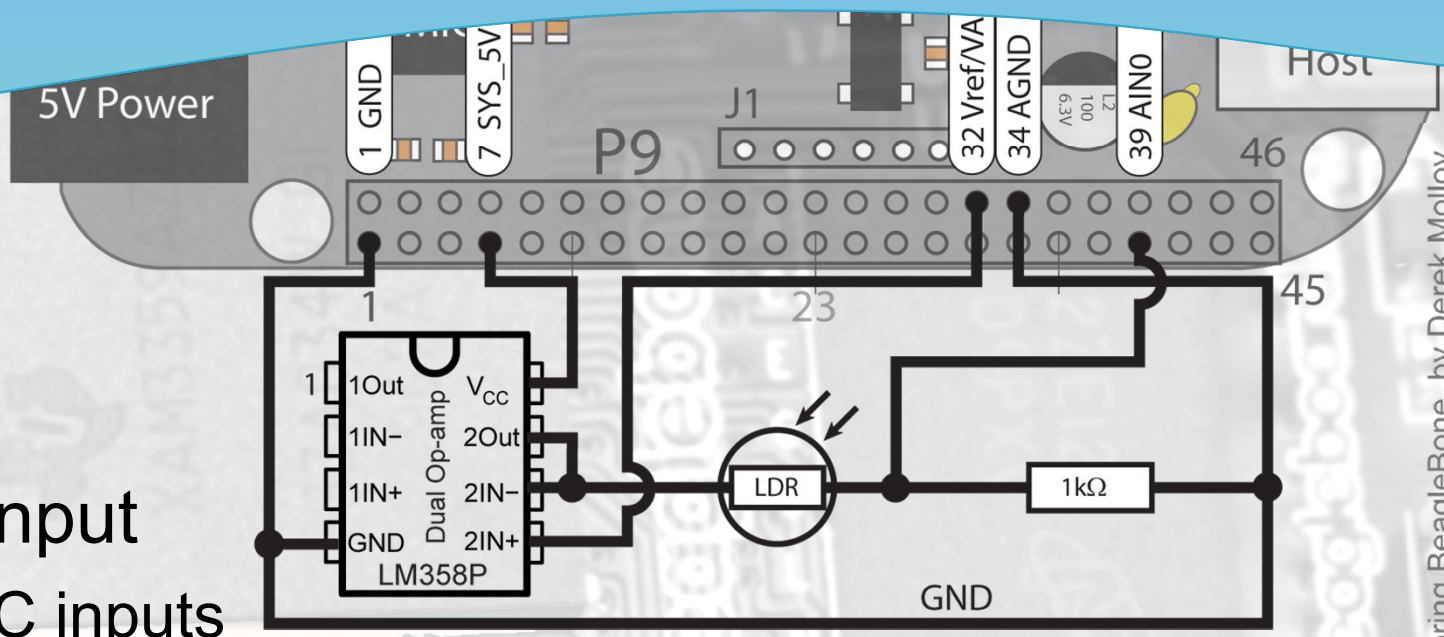




- C/C++ Linux userspace code available
  - Must configure internal resistor characteristics
  - Response latency as low as 324µS in Linux
  - GPIO-Keys allows for generalized interface
  - Can directly memory switch (dangerous?)
  - Can use the PRU-ICSS

Image from Exploring BeagleBone, by Derek Molloy

Interfacing to the BBB
Analogue Input



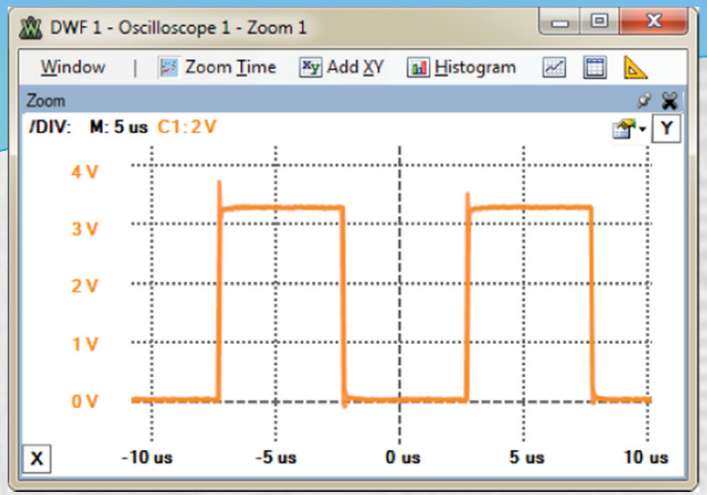
- Analog Input
  - 7 x ADC inputs
  - Pins 1.8V tolerant – use Vref (not a supply!)
  - 12-bit SAR ADC 200,000S/s (8:1 switch)
  - Load device tree overlay, C/C++ code available

Image from Exploring BeagleBone, by Derek Molloy

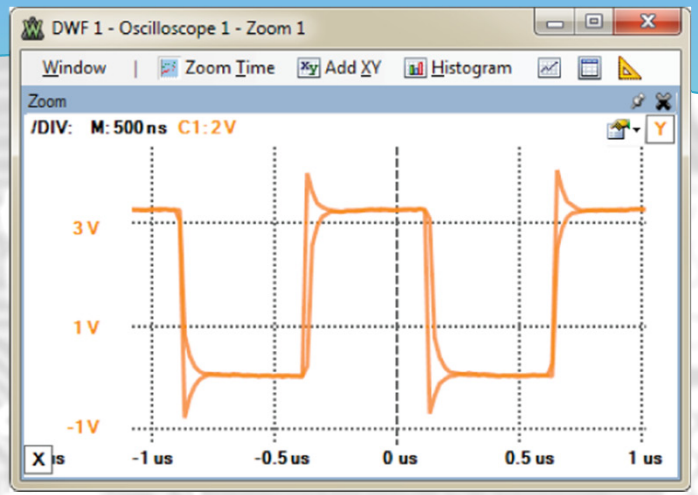
Interfacing to the BBB

PWM Output

50% @ 100 kHz



50% @ 1 MHz



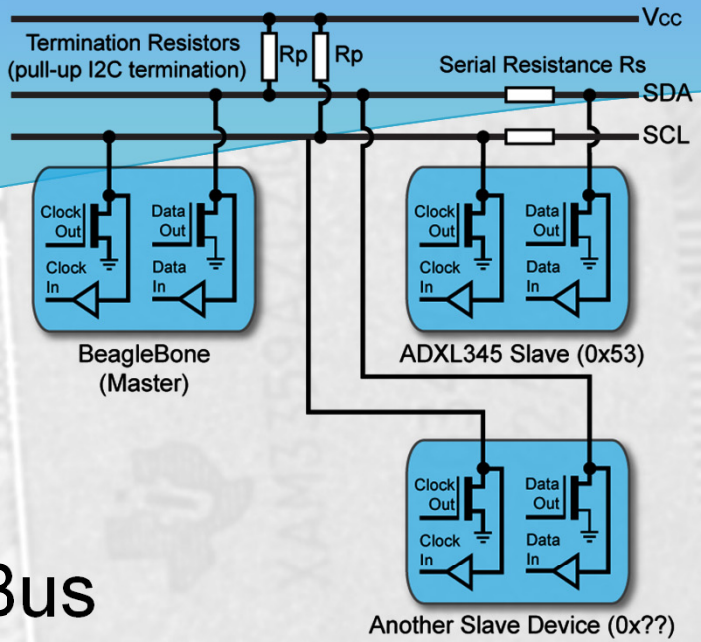
- Pulse Width Modulation (PWM) Outputs
  - 14 x PWM (configurable from Linux userspace)

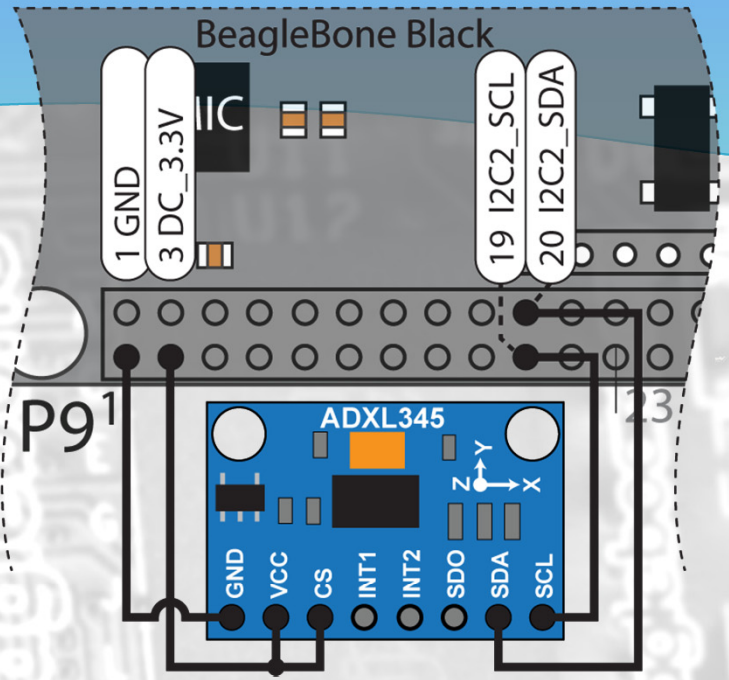
```
/sys/devices/ocp.3/pwm_test_P9_22.15$ sudo su
/sys/devices/ocp.3/pwm_test_P9_22.15# echo 5000 > duty
/sys/devices/ocp.3/pwm_test_P9_22.15# echo 10000 > period
/sys/devices/ocp.3/pwm_test_P9_22.15# echo 1 > run
```

Image from Exploring BeagleBone, by Derek Molloy

Interfacing to the BBB

I²C Interface





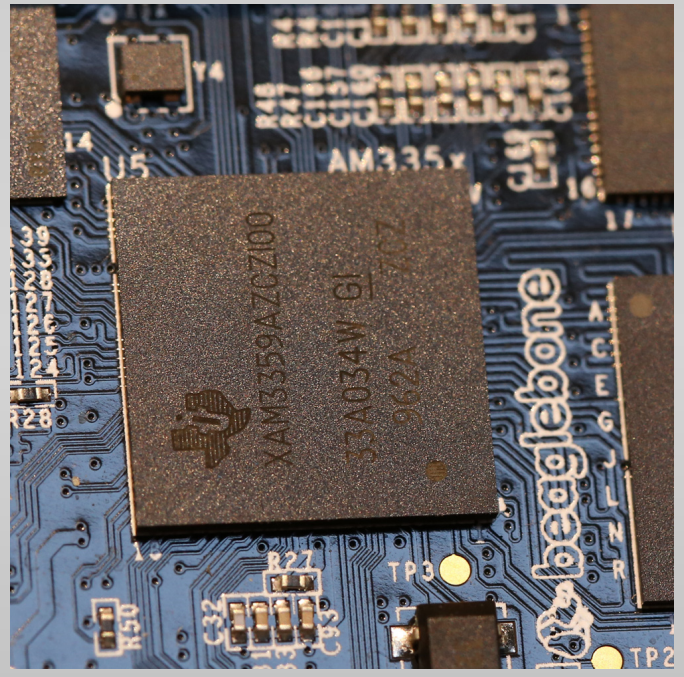
- I²C Bus
  - 2 x public, 1 x private
  - 3.3V bus – be careful of logic-level translation
  - Have Linux userspace tools (I2C-Tools)
  - Can use Linux ioctl calls to control the bus in C/C++

Image from Exploring BeagleBone, by Derek Molloy

Overview

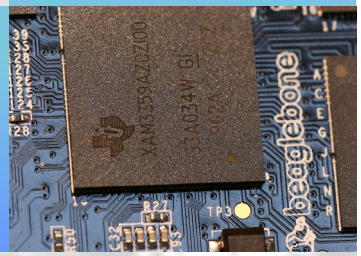
The BeagleBone: Application in Engineering Education

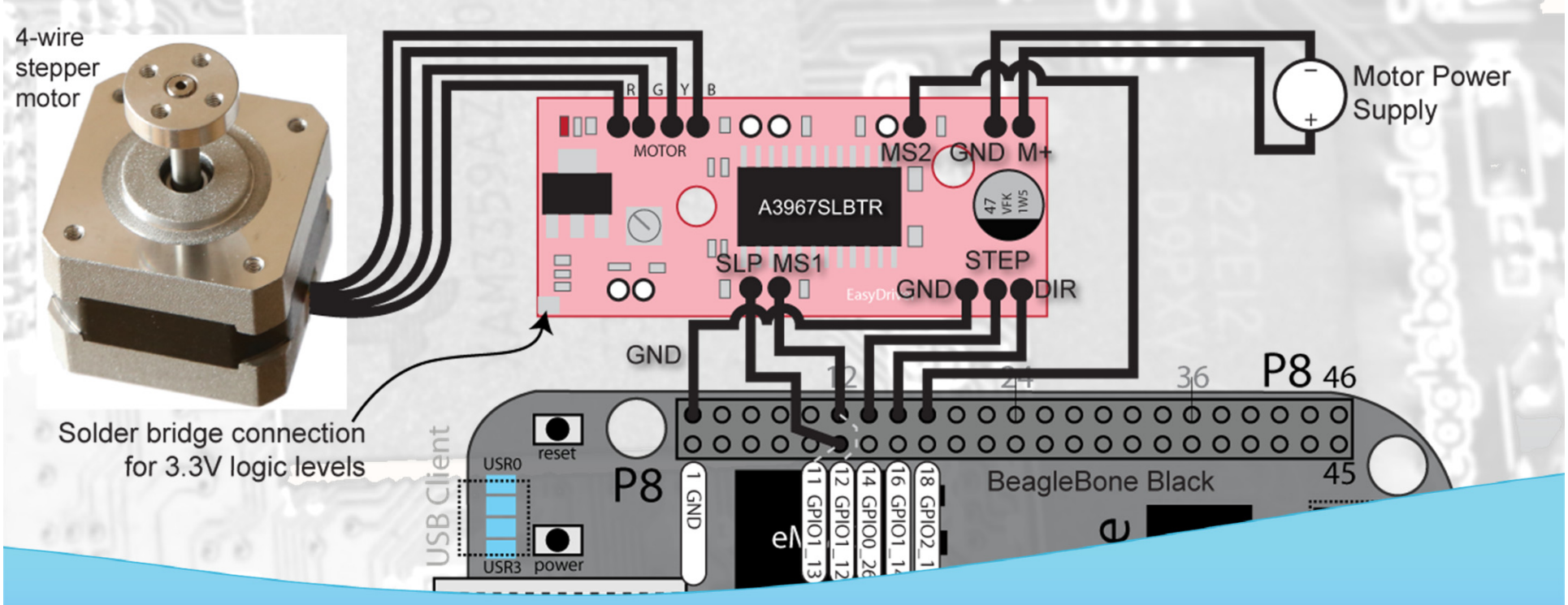
1. Introduction
2. BeagleBone Black Hardware
3. BeagleBone Black Software
4. Software Development Tools
5. Interfacing to the BBB
6. ✓ **Applications**
7. Real-Time BeagleBone
8. Conclusions



Interfacing Applications

Connecting to Motors using GPIOs

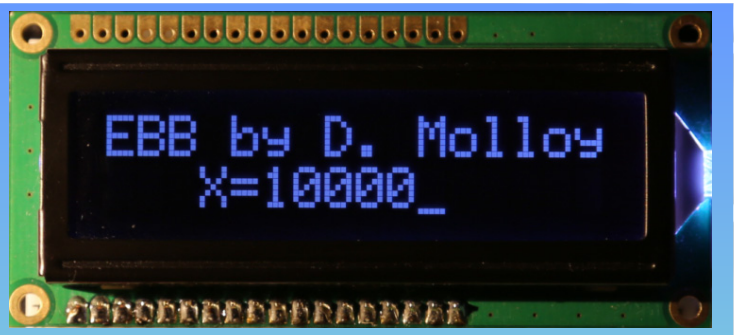




- Motor boards (e.g., TI DRV8835), stepper boards
- Interface to servo motors using PWM pins
- Can integrate Posix threads, wrap with classes

Interfacing Applications

Analogue Sensors



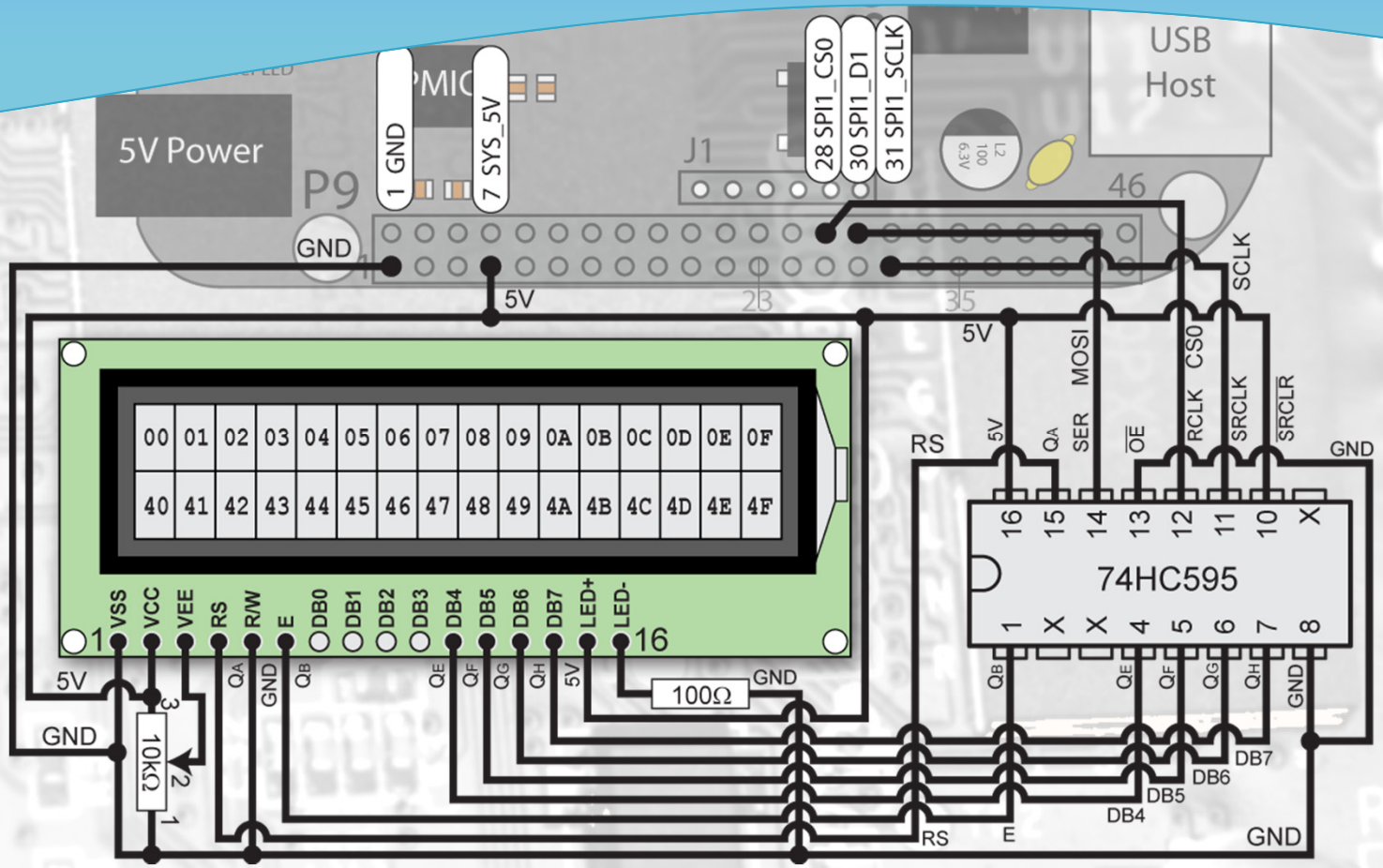


Image from Exploring BeagleBone, by Derek Molloy

Interfacing Applications

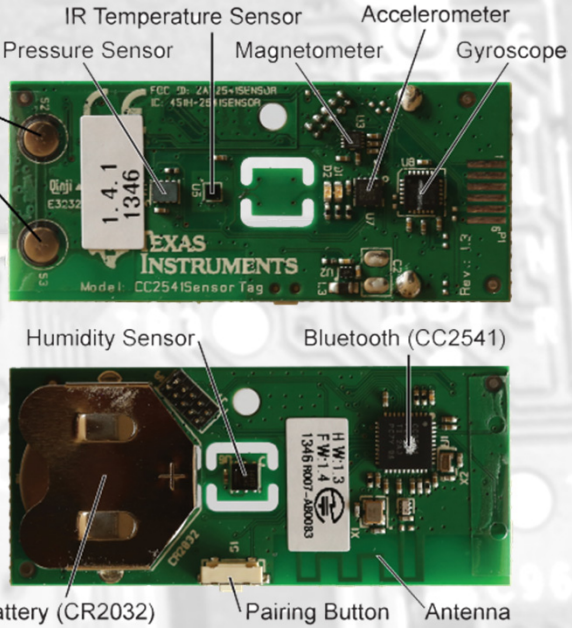
Analogue Sensors

2.4 GHz, 30ft+ range, low power

Desktop Computer with Bluetooth Adapter
(RealTerm Serial Capture)
Applications: Wireless control, Serial control, Desktop audio, File Sharing

Phone/Tablet Computer
(Google Play - Bluetooth Module CI)
Applications: Robot control, diagnostics, sensor calibration

BeagleBone



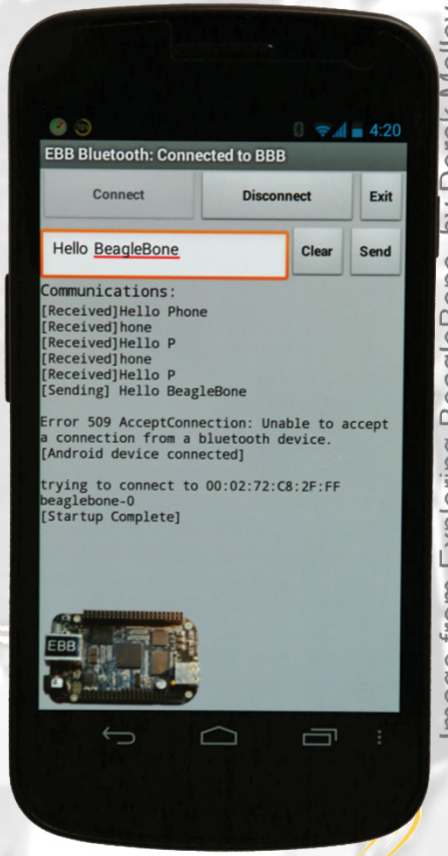


Image from Exploring BeagleBone, by Derek Molloy

Interfacing Applications
Platform as a Service (PaaS)

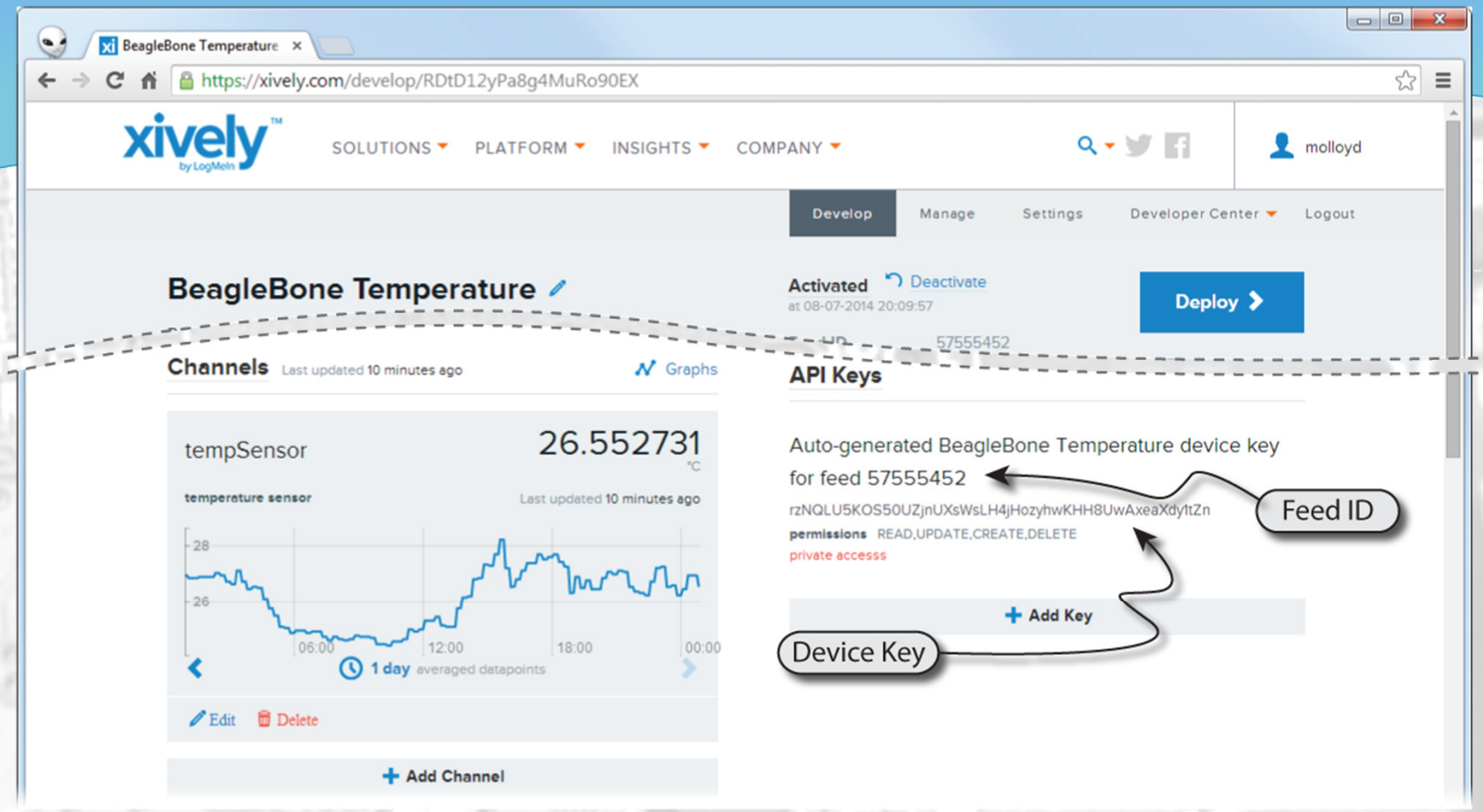


Image from Exploring BeagleBone, by Derek Molloy

Interfacing Applications

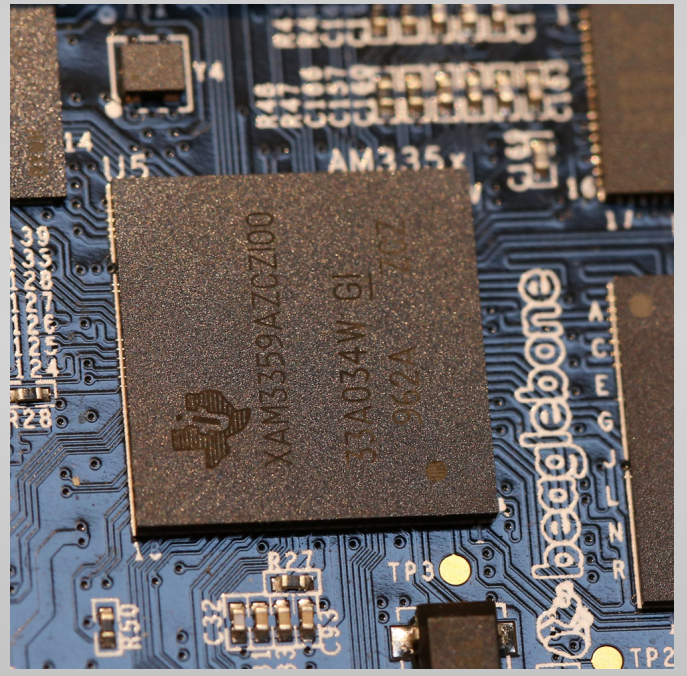
Video, image processing, computer vision

Audio input, output and streaming

Image from Exploring BeagleBone, by Derek Molloy

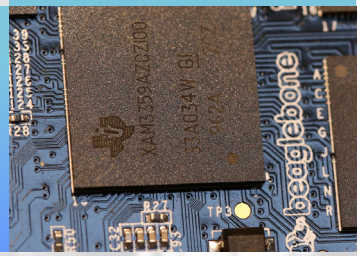
Overview

The BeagleBone: Application in Engineering Education



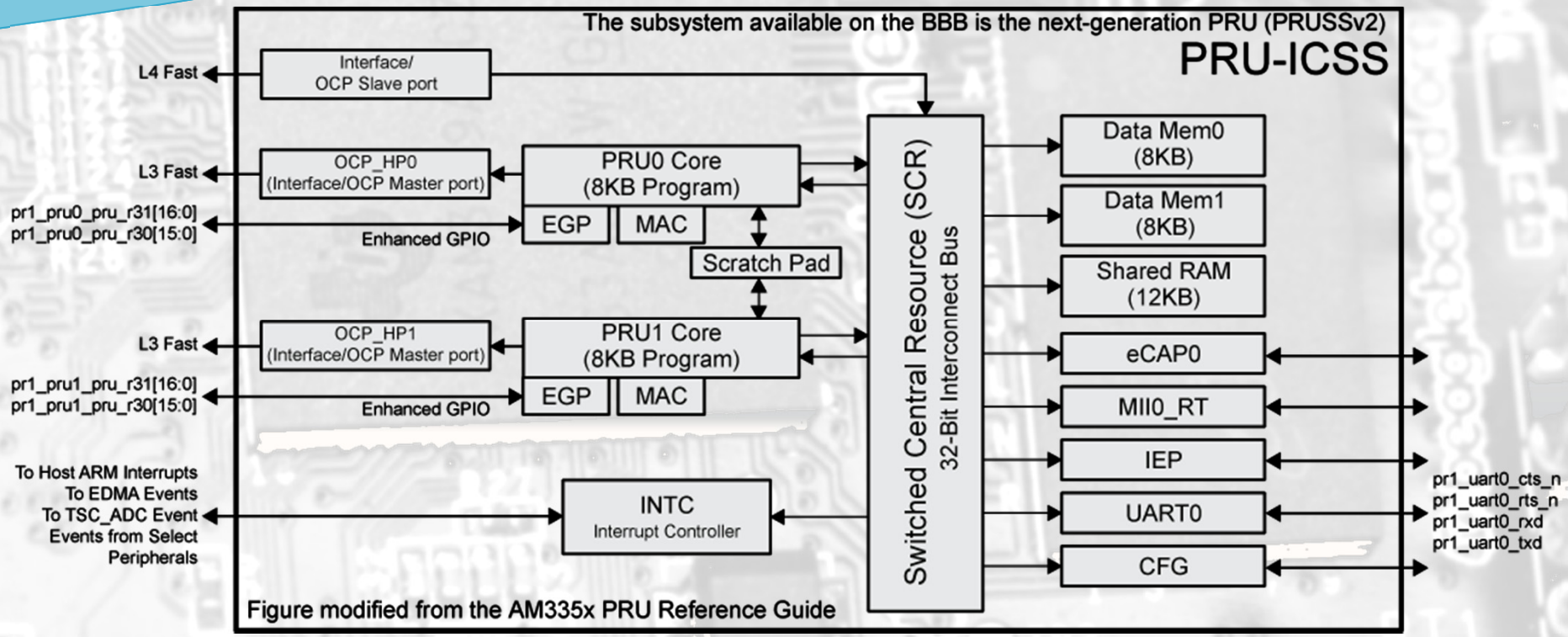
Real-time BeagleBone

Real-time on the BBB



- Limitations of non-preemptive Linux…
- Real-Time Kernels (preemptive Linux)
  - RT-Preempt patch (PREEMPT_RT)
  - Xenomai co-kernel
- Interface BBB to Stellaris (e.g., using UART)
- Use the AM3358 PRU-ICSS
  - Programmable real-time units and industrial communication sub-system
  - 2 x PRUs @ 400 MHz, 5ns per instruction
  - small RISC ISA (~45 instructions)
  - 8KB instruction memory + 8KB data RAM for each PRU
  - 12KB shared memory and **full access to Linux memory space**

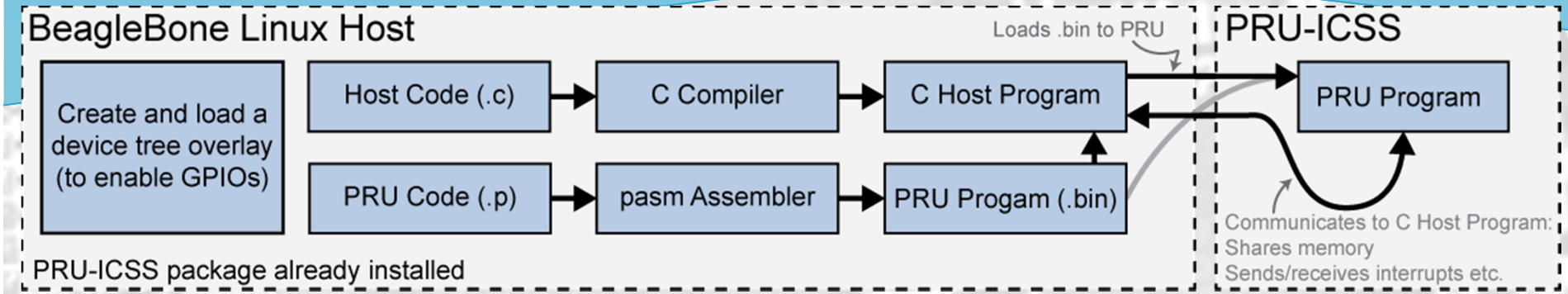
Real-time BeagleBone
Architecture



Customized for the BBB from an image that is courtesy of Texas Instruments

Real-Time BeagleBone

Deploying Program to one of the PRUs



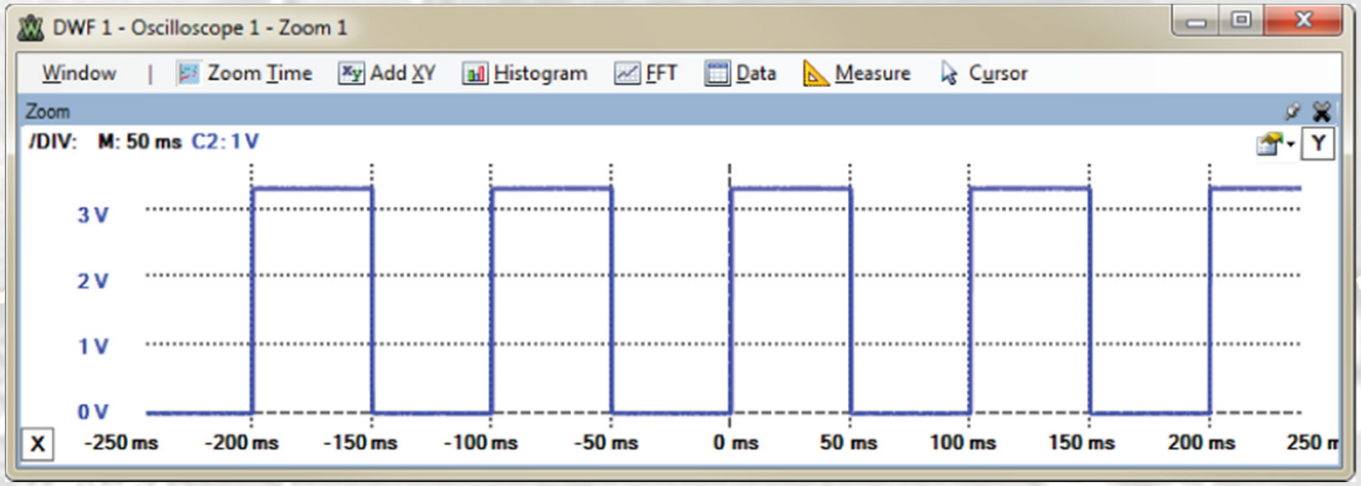
- PRU has access to Enhanced GPIOs
  - Use device tree overlay to set up the PRU EGPIOs
  - Access to Linux memory slower than PRU local memory
- C Host program loads PRU binary into PRU
  - Shares memory with the host (PRU memory mapped to userspace)
  - Use interrupts to trigger events

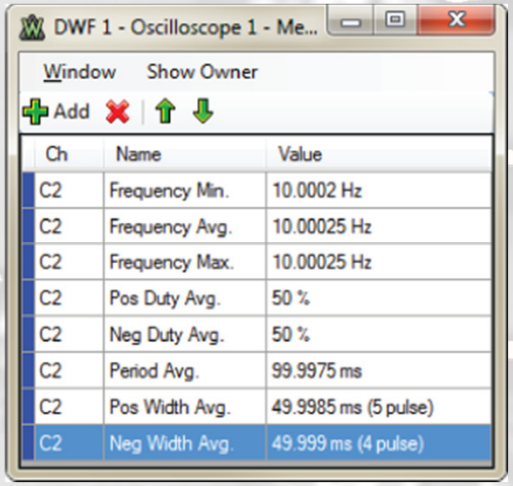
ledButton.p

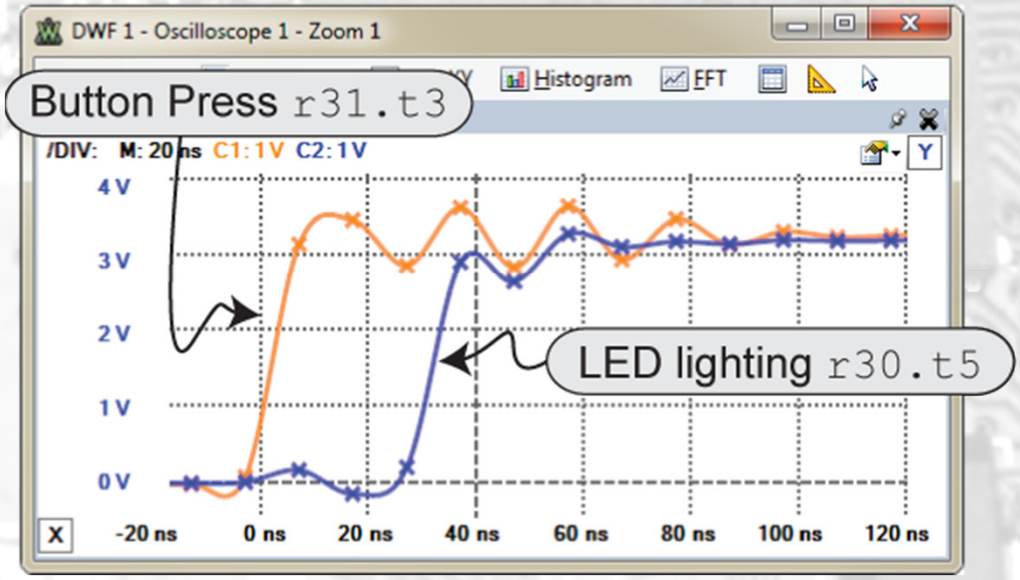
```
.origin 0                           // start of program in PRU memory
.entrypoint START                   // program entry point (for a debugger)

#define INS_PER_US          200     // 5ns per instruction
#define INS_PER_DELAY_LOOP 2        // two instructions per delay loop
                                    // set up a 50ms delay
#define DELAY  50 * 1000 * (INS_PER_US / INS_PER_DELAY_LOOP)
#define PRU0_R31_VEC_VALID 32       // allows notification of program completion
#define PRU_EVTOUT_0        3       // the event number that is sent back

START:
        SET     r30.t5              // turn on the output pin (LED on)
        MOV     r0, DELAY           // store the length of the delay in REG0
DELAYON:
        SUB     r0, r0, 1           // Decrement REG0 by 1
        QBNE    DELAYON, r0, 0      // Loop to DELAYON, unless REG0=0
LEDOFF:
        CLR     r30.t5              // clear the output bin (LED off)
        MOV     r0, DELAY           // Reset REG0 to the length of the delay
DELAYOFF:
        SUB     r0, r0, 1           // decrement REG0 by 1
        QBNE    DELAYOFF, r0, 0     // Loop to DELAYOFF, unless REG0=0
        QBBC    START, r31.t3       // is the button pressed? If not, loop
END:                                // notify the calling app that finished
        MOV     r31.b0, PRU0_R31_VEC_VALID | PRU_EVTOUT_0
        HALT                        // halt the pru program
```

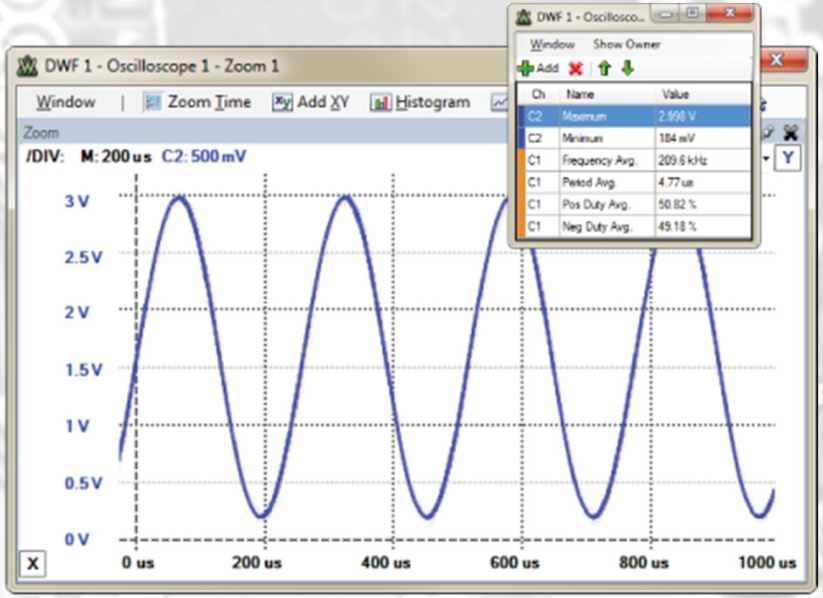
ledButton.c

```
#include <stdio.h>
#include <prussdrv.h>
#include <pruss_intc_mapping.h>
#define PRU_NUM 0   // using PRU0 for these examples

void main (void)
{
   // Initialize structure used by prussdrv_pruintc_intc
   // PRUSS_INTC_INITDATA is found in pruss_intc_mapping.h
   tpruss_intc_initdata pruss_intc_initdata = PRUSS_INTC_INITDATA;
   // Allocate and initialize memory
   prussdrv_init ();
   prussdrv_open (PRU_EVTOUT_0);
   // Map PRU's interrupts
   prussdrv_pruintc_init(&pruss_intc_initdata);
   // Load and execute the PRU program on the PRU
   prussdrv_exec_program (PRU_NUM, "./ledButton.bin");
   // Wait for event completion from PRU, returns the PRU_EVTOUT_0 number
   int n = prussdrv_pru_wait_event (PRU_EVTOUT_0);
   printf("EBB PRU program completed, event number %d.\n", n);
   // Disable PRU and close memory mappings
   prussdrv_pru_disable(PRU_NUM);
   prussdrv_exit();
}
```


DWF 1 - Oscilloscope 1 - Zoom 1
Window
Zoom Time
Add XY
Histogram
FFT
Data
Measure
Cursor
Zoom
/DIV: M: 50 ms C2: 1 V
3 V
2 V
1 V
0 V
-250 ms
-200 ms
-150 ms
-100 ms
-50 ms
0 ms
50 ms
100 ms
150 ms
200 ms


DWF 1 - Oscilloscope 1 - Me...
Window
Show Owner
Add
Ch
Name
Value
C2 Frequency Min. 10.0002 Hz
C2 Frequency Avg. 10.00025 Hz
C2 Frequency Max. 10.00025 Hz
C2 Pos Duty Avg. 50 %
C2 Neg Duty Avg. 50 %
C2 Period Avg. 99.9975 ms
C2 Pos Width Avg. 49.9985 ms (5 pulse)
C2 Neg Width Avg. 49.999 ms (4 pulse)

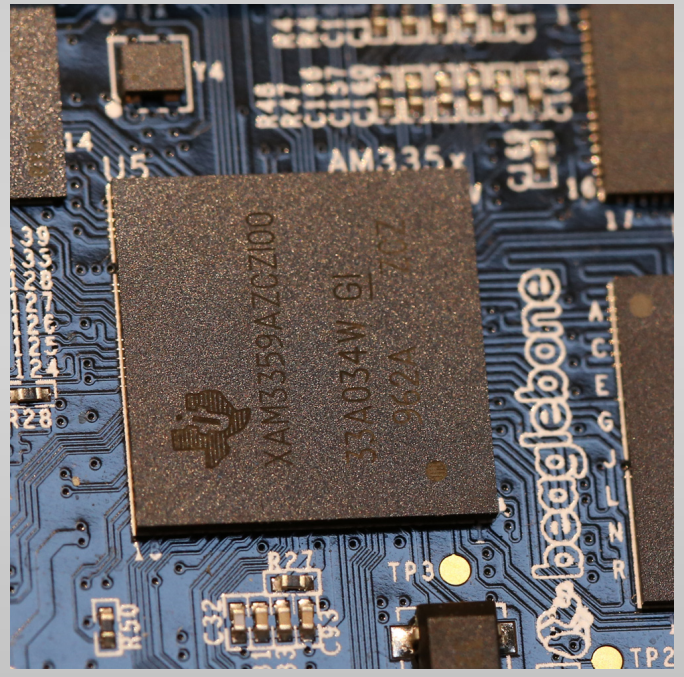

DWF 1 - Oscilloscope 1 - Zoom 1
Button Press r31.t3
Histogram
FFT
/DIV: M: 20 ns C1: 1 V C2: 1 V
4 V
3 V
2 V
1 V
0 V
-20 ns
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
LED lighting r30.t5


DWF 1 - Oscilloscope 1 - Zoom 1
Window
Zoom Time
Add XY
Histogram
Zoom
/DIV: M: 200 us C2: 500 mV
3 V
2.5 V
2 V
1.5 V
1 V
0.5 V
0 V
0 us
200 us
400 us
600 us
800 us
1000 us

Overview

The BeagleBone: Application in Engineering Education

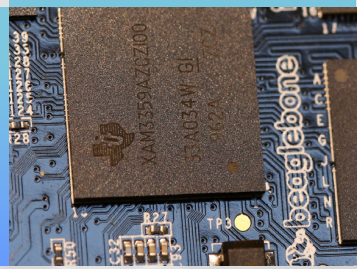
1. Introduction
2. BeagleBone Black Hardware
3. BeagleBone Black Software
4. Software Development Tools
5. Interfacing to the BBB
6. Applications
7. Real-Time BeagleBone
8. ✓ **Conclusions**



Conclusions

Positives and Negatives of the BBB in Education

Positives	Negatives
Low-cost per unit	Supply of boards constrained
Practical integration of electronics, software and Linux	Difficult to support range of software/hardware issues that can arise – especially corporate laptops!
Exposure to embedded Linux	Embedded Linux is a moving target
USB-over-Internet on-campus	Device tree overlays complex
Easy to burn new image to eMMC	
Great for project work	



Google+:
google.com/+DerekMolloy
E-mail:
derek.molloy@dcu.ie
Blog/Website:
www.derekmolloy.ie

~ 576 pages
Describes everything in this presentation (properly!)

~ December 2014
Available for pre-order on Amazon:
~ $24.92
~ £20.68
~ €25.84

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265