CEC Continuing Education Center

DesignNews

C++ Primer for Embedded Systems

Day 3:

Embedded C++: The Basics

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Fred Eady

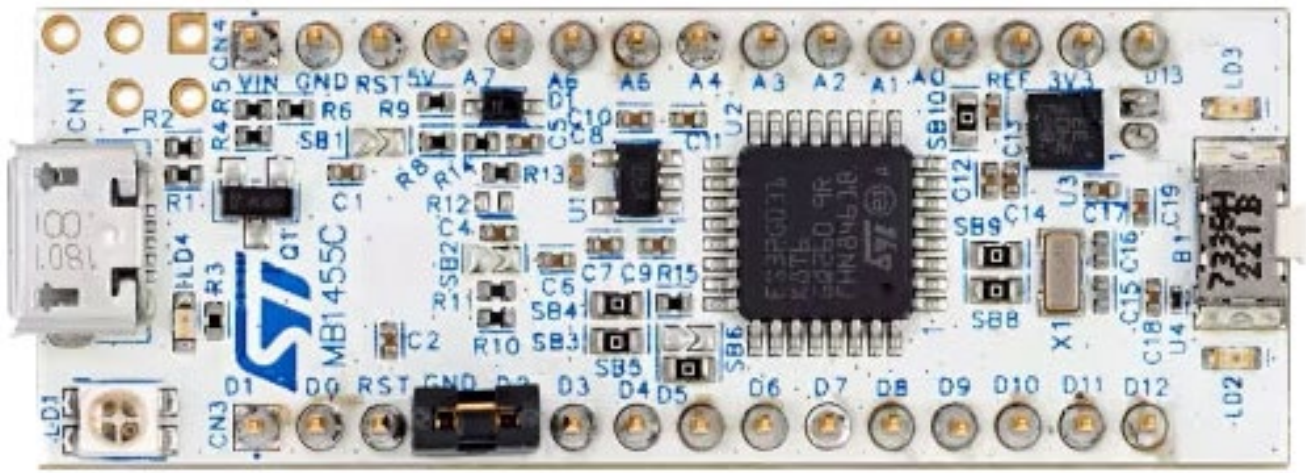
Visit ‘Lecturer Profile’ in your console for more details.

CEC Continuing Education Center

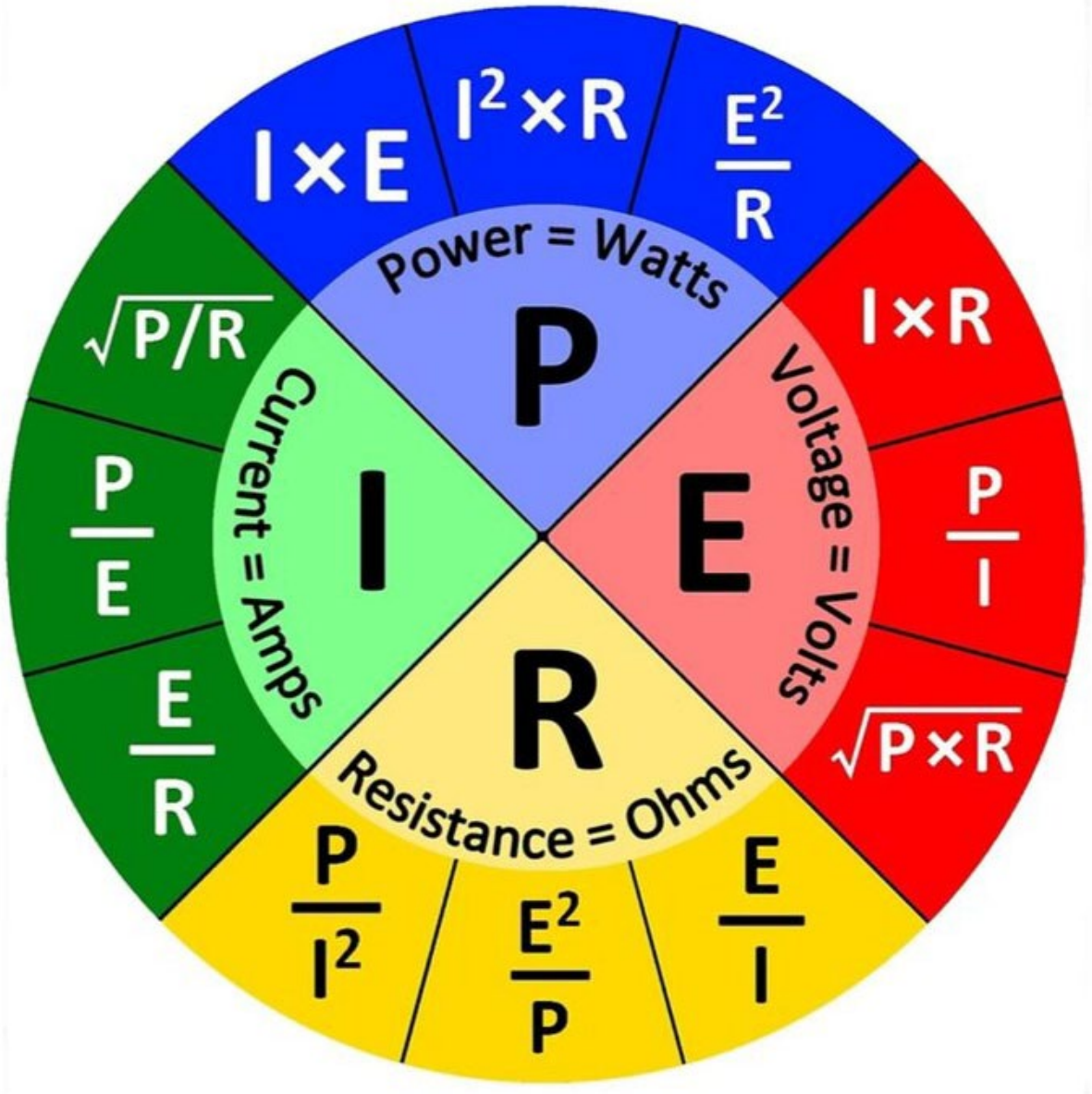
AGENDA

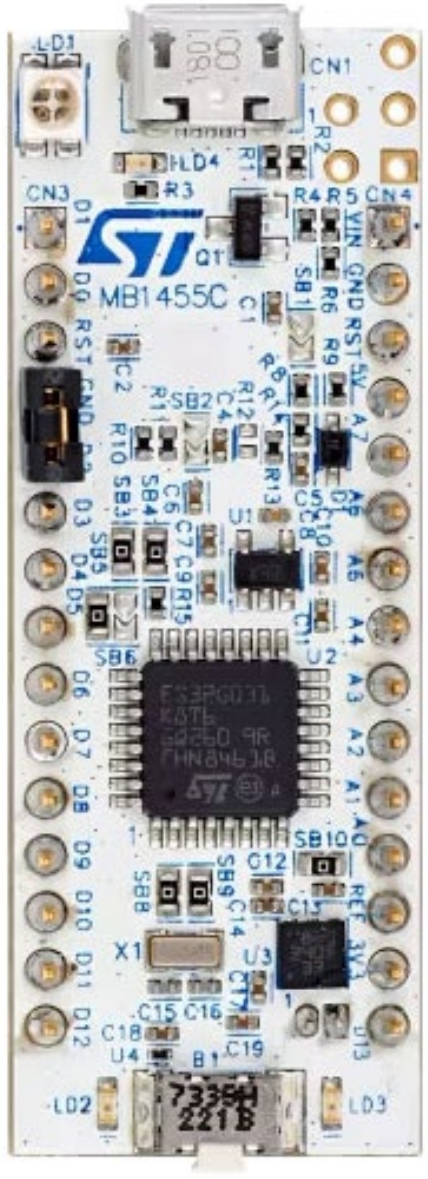
DigiKey

- C++ OOP Components
  - Class
  - Encapsulation
  - Abstraction
  - Constructor
  - Inheritance
  - Polymorphism
- Enable printf float
- Redirect the UART



C++ Class Components





C++ Class Components

```
63  class OHMSLAW                          // Class Keyword - Class Name
64  {
65      public:                            // Access specifier - Members are accessible outside of the class
66          OHMSLAW(float e, float i, float r);   // Constructor - Same name as the Class with initializer list
67
68          void V();                      // Public Member - Methods (Functions)
69          void I();
70          void R();
71
72      private:                           // Access specifier - Members are not accessible outside of the class
73          float volts;                   // Private Member - Attributes
74          float current;
75          float resistance;
76
77      protected:                         // Access specifier - Members are accessible inside of class and derived classes
78  };
```

C++ Encapsulation

```
63  class OHMSLAW                       // Class Keyword - Class Name
64  {
65      public:                         // Access specifier - Members are accessible outside of the class
66          OHMSLAW(float e, float i, float r);   // Constructor - Same name as the Class with initializer list
67
68          void V();                   // Public Member - Methods (Functions)
69          void I();
70          void R();
71
72      private:                        // Access specifier - Members are not accessible outside of the class
73          float volts;                // Private Member - Attributes
74          float current;
75          float resistance;
76
77      protected:                      // Access specifier - Members are accessible inside of class and derived classes
78  };
```

Encapsulation within the Class

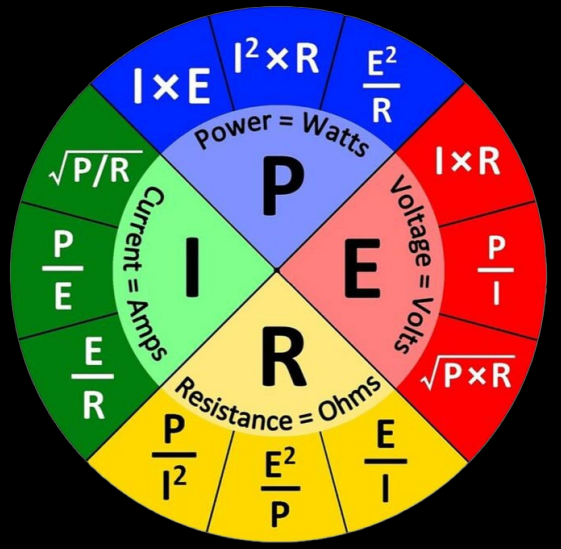
C++ Abstraction

```
63  class OHMSLAW                          // Class Keyword - Class Name
64  {
65    public:                              // Access specifier - Members are accessible outside of the class
66      OHMSLAW(float e, float i, float r);// Constructor - Same name as the Class with initializer list
67
68      void V();                          // Public Member - Methods (Functions)
69      void I();
70      void R();
71
72    private:                             // Access specifier - Members are not accessible outside of the class
73      float volts;                       // Private Member - Attributes
74      float current;
75      float resistance;
76
77    protected:                           // Access specifier - Members are accessible inside of class and derived classes
78  };
```

Data Hiding Using Private Attributes Enables Abstraction

C++ Constructor and Methods – Defined Outside of the Class

```
80  // This Constructor will be called every time an OHMSLAW object is created
81  OHMSLAW::OHMSLAW(float e, float i, float r) // Constructor parameter list
82  |  :                                       // Parameter list/Member Initializer Delimiter
83  |  volts(e),                               // Member Initializer list
84  |  current(i),
85  |  resistance(r)
86  {}                                         // Constructor Body - You can put code here
87
88  // The :: in each function declaration is known as the scope resolution operator
89  void OHMSLAW::V()
90  {
91  |  //volts = current * resistance;
92  |  printf("volts = %3.2f\r\n",current * resistance);
93  }
94
95  void OHMSLAW::I()
96  {
97  |  //current = volts / resistance;
98  |  printf("current = %3.2f\r\n",volts / resistance);
99  }
100
101 void OHMSLAW::R()
102 {
103 |  //resistance =  volts / current;
104 |  printf("resistance = %3.2f\r\n",volts / current);
105 }
```



C++ Abstraction Using the OHMSLAW Class

```
141  // Create OHMSLAW objects calcVoltage, calcCurrent and calcResistance
142  OHMSLAW calcVoltage(0,1.32,2.00);     // Invoke the OHMSLAW Constructor and initialize private attributes e,i,r
143  calcVoltage.V();                      // Execute the OHMSLAW V() Method
144  OHMSLAW calcCurrent(4.12,0,2.67);     // Invoke the OHMSLAW Constructor and initialize private attributes e,i,r
145  calcCurrent.I();                      // Execute the OHMSLAW I() Method
146  OHMSLAW calcResistance(1.67,1.87,0);  // Invoke the OHMSLAW Constructor and initialize private attributes e,i,r
147  calcResistance.R();                   // Execute the OHMSLAW R() Method
148
149
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

+ Open an additional monitor

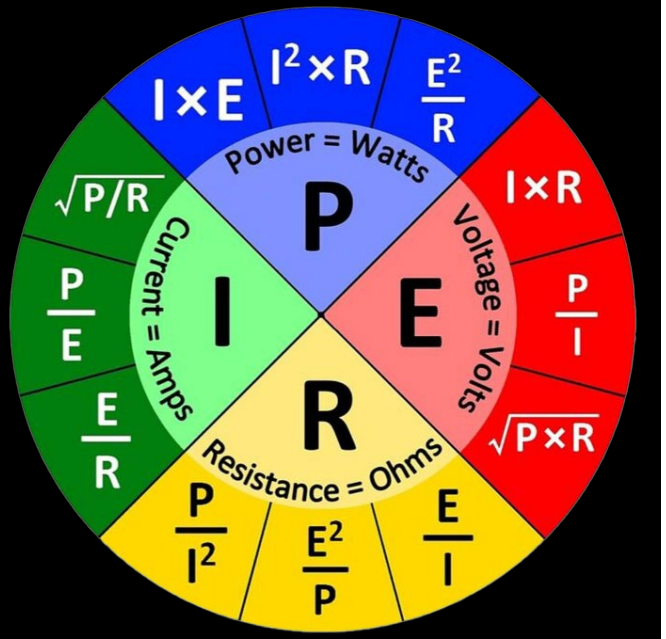
Monitor Mode: Serial | View Mode: Text | Port: /dev/ttyACM0 - STMicroelectronics | Baud rate: 115200 | Line ending: None

Stop Monitoring

```
volts = 2.64
current = 1.54
resistance = 0.89
```

C++ Constructor and Methods – Defined Inside of the Class

```
72  class OHMSLAW                          // Class Keyword - Class Name
73  {
74    public:                              // Access specifier - Members accessible outside of the class
75      OHMSLAW(float e, float i, float r) // Constructor - Same name as the Class with parameter list
76      :                                  // Parameter list/Member Initializer Delimiter
77      volts(e),                          // Member Initializer list
78      current(i),
79      resistance(r)
80    {}                                   // Constructor Body - You can put code here
81
82      // Public Member - Methods (Functions)
83      void V()
84      {
85        //volts = current * resistance;
86        printf("volts = %3.2f\r\n",current * resistance);
87      }
88
89      void I()
90      {
91        //current = volts / resistance;
92        printf("current = %3.2f\r\n",volts / resistance);
93      }
94
95      void R()
96      {
97        //resistance =  volts / current;
98        printf("resistance = %3.2f\r\n",volts / current);
99      }
100
101   private:                               // Access specifier - Members not accessible outside of the class
102     float volts;                         // Private Member - Attributes
103     float current;
104     float resistance;
105
106   protected:                             // Access specifier - Members accessible inside of class and derived classes
107 };
```



C++ Abstraction Using the OHMSLAW Class

```
135  // Create OHMSLAW objects calcVoltage, calcCurrent and calcResistance
136  OHMSLAW calcVoltage(0,1.38,1.15);     // Invoke the OHMSLAW Constructor and initialize
137  calcVoltage.V();                      // Execute the OHMSLAW V() Method
138  OHMSLAW calcCurrent(1.2,0,1.75);      // Invoke the OHMSLAW Constructor and initialize
139  calcCurrent.I();                      // Execute the OHMSLAW I() Method
140  OHMSLAW calcResistance(1.32,1.29,0);  // Invoke the OHMSLAW Constructor and initialize
141  calcResistance.R();                   // Execute the OHMSLAW R() Method
142
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

+ Open an additional monitor

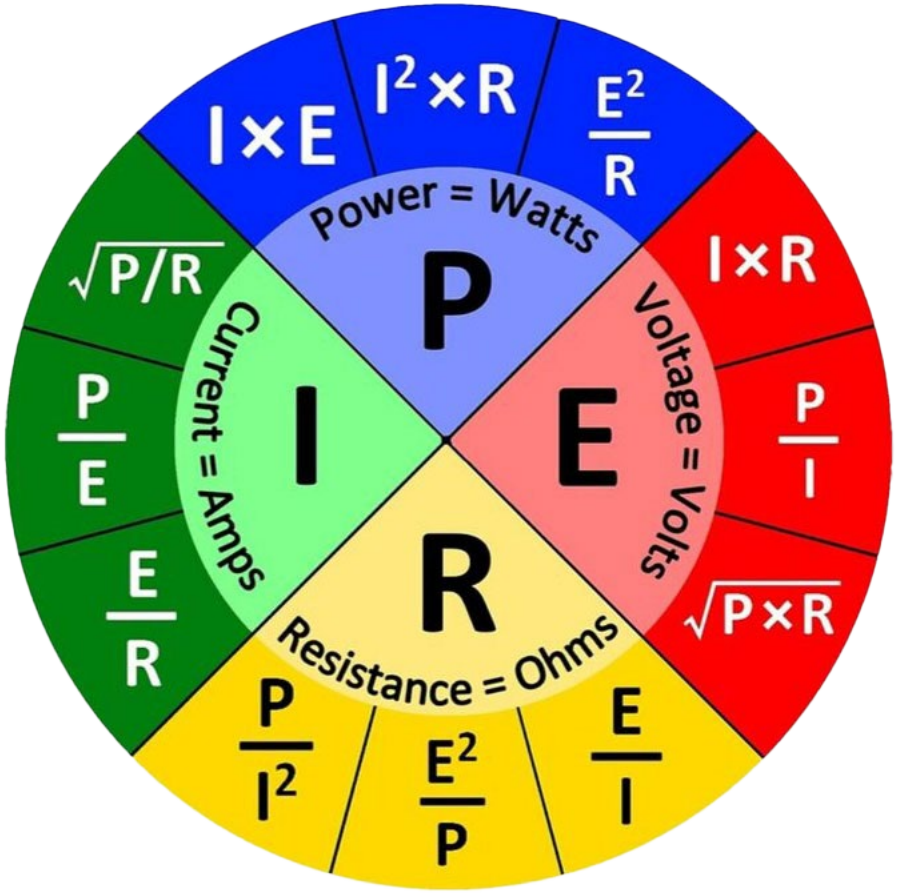
Monitor Mode: Serial | View Mode: Text | Port: /dev/ttyACM0 - STMicroelectronics | Baud rate: 115200 | Line ending: None

Stop Monitoring

```
volts = 1.59
current = 0.69
resistance = 1.02
```

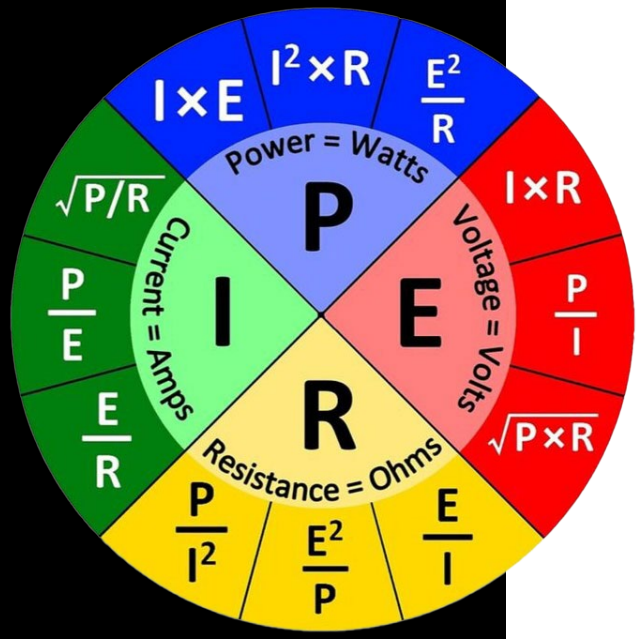
C++ Inheritance – Parent OHMSLAW Class

```
71  class OHMSLAW                          // Class Keyword - Class Name
72  {
73    public:
74      // Public Member - Methods (Functions)
75      void V()
76      {
77        //volts = current * resistance;
78        printf("volts = %3.2f\r\n",current * resistance);
79      }
80
81      void I()
82      {
83        //current = volts / resistance;
84        printf("current = %3.2f\r\n",volts / resistance);
85      }
86
87      void R()
88      {
89        //resistance =  volts / current;
90        printf("resistance = %3.2f\r\n",volts / current);
91      }
92
93      float volts;                       // Public Member - Attributes
94      float current;
95      float resistance;
96  };
```



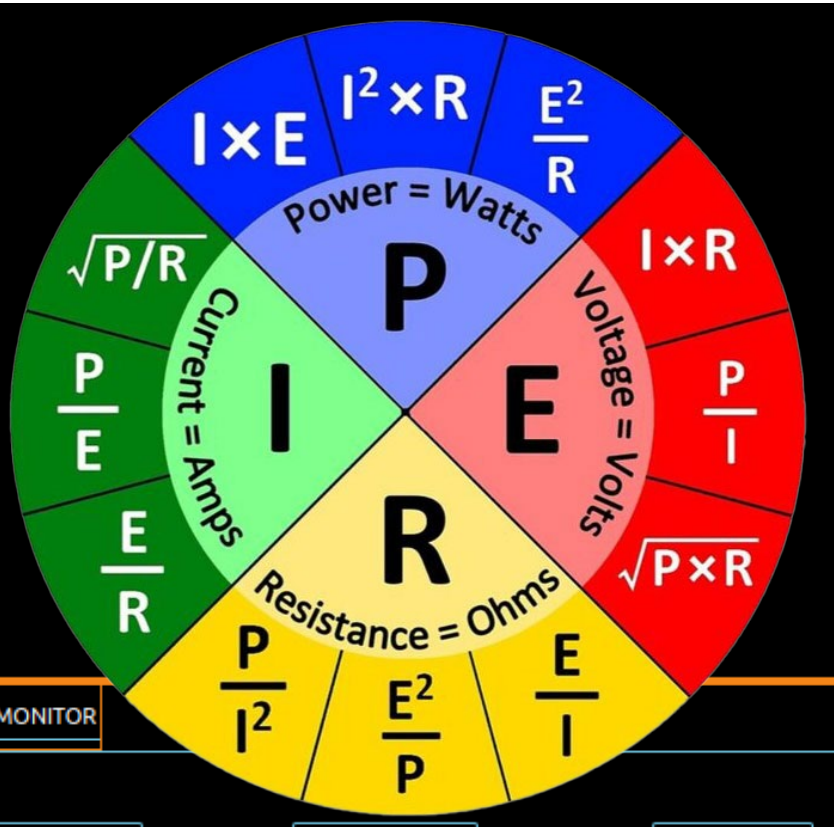
C++ Inheritance – Child WATTSLAW Class

```
77  class WATTSLAW: public OHMSLAW
78  {
79    public:
80    WATTSLAW(float e, float i, float r)  // Constructor - Same name as the Class with parameter list
81    :                                    // Parameter list/Member Initializer Delimiter
82    wvolts(e),                           // Member Initializer list
83    wcurrent(i),
84    wresistance(r)
85  {}                                     // Constructor Body - You can put code here
86
87    float wvolts;
88    float wcurrent;
89    float wresistance;
90
91    void P_EI()
92    {
93      printf("power = %3.2f\r\n",wvolts * wcurrent);
94    }
95
96    void P_I2R()
97    {
98      printf("power = %3.2f\r\n",(wcurrent * wcurrent) * wresistance);
99    }
100
101   void P_E2R()
102   {
103     printf("power = %3.2f\r\n",(wvolts * wvolts) / wresistance);
104   }
105
106 };
```



C++ Inheritance

```
163  WATTSLAW calcPower(2,3,0.66);  // e,i,r
164  calcPower.P_EI();
165  calcPower.P_I2R();
166  calcPower.P_E2R();
167
168  // Method and Attributes of the OHMSLAW Parent Class
169  calcPower.resistance = 220;
170  calcPower.current = 0.010;
171  calcPower.V();
172
173  calcPower.volts = 3.300;
174  calcPower.resistance = 0.229;
175  calcPower.I();
176
177  calcPower.volts = 5.25;
178  calcPower.current = 0.002;
179  calcPower.R();
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

+ Open an additional monitor

Monitor Mode: Serial | View Mode: Text | Port: /dev/ttyACM0 - STMicroelectronics | Baud rate: 115200 | Line ending: None

Stop Monitoring

```
power = 6.00
power = 5.94
power = 6.06
volts = 2.20
current = 14.41
resistance = 2625.00
```

C++ Polymorphism

```
 95  class POLY
 96  {
 97    public:
 98      void prtnum(uint8_t num1)
 99      {
100        printf("%u\r\n",num1);
101      }
102
103      void prtnum(uint8_t num1, uint8_t num2)
104      {
105        printf("%u - %u\r\n",num1,num2);
106      }
107
108      void prtnum(uint8_t num1, uint8_t num2, uint8_t num3)
109      {
110        printf("%u - %u - %u\r\n",num1,num2,num3);
111      }
112
113  };
```

C++ Polymorphism

```
115   POLY numobject;
116   numobject.prtnum(1);
117   numobject.prtnum(1,2);
118   numobject.prtnum(1,2,3);
```

PROBLEMS 1 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | MEMORY | XRTOS | SERIAL MONITOR

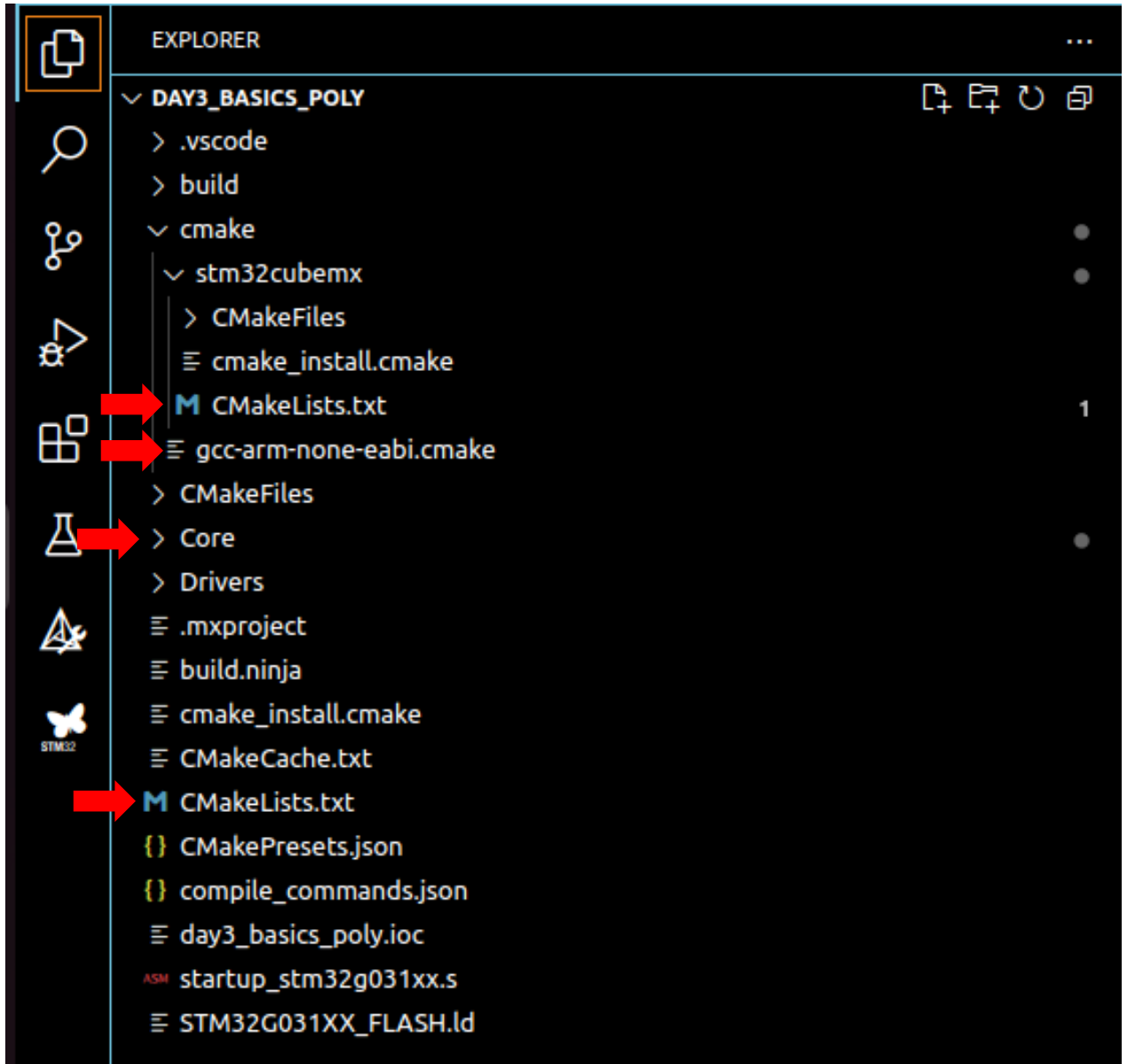
+ Open an additional monitor

Monitor Mode: Serial | View Mode: Text | Port: /dev/ttyACM0 - STMicroelectronics | Baud rate: 115200 | Line ending: None

Stop Monitoring

```
1
1 - 2
1 - 2 - 3
```

Enable printf float – Redirect UART



Enable printf float – Modify gcc-arm-none-eabi.make

gcc-arm-none-eabi.cmake

cmake > gcc-arm-none-eabi.cmake

```
38  set(CMAKE_C_LINK_FLAGS "${TARGET_FLAGS}")
39  set(CMAKE_C_LINK_FLAGS "${CMAKE_C_LINK_FLAGS} -T \"${CMAKE_SOURCE_DIR}/STM32G031XX_FLASH.ld\"")
40  set(CMAKE_C_LINK_FLAGS "${CMAKE_C_LINK_FLAGS} --specs=nano.specs")
41  set(CMAKE_C_LINK_FLAGS "${CMAKE_C_LINK_FLAGS} -Wl,-Map=${CMAKE_PROJECT_NAME}.map -Wl,--gc-sections")
42  set(CMAKE_C_LINK_FLAGS "${CMAKE_C_LINK_FLAGS} -Wl,--start-group -lc -lm -Wl,--end-group")
43  set(CMAKE_C_LINK_FLAGS "${CMAKE_C_LINK_FLAGS} -Wl,--print-memory-usage")
44  # Add Line below to enable printf float
45  set(CMAKE_C_LINK_FLAGS "${CMAKE_C_LINK_FLAGS} -u _printf_float")
46
47  set(CMAKE_CXX_LINK_FLAGS "${CMAKE_C_LINK_FLAGS} -Wl,--start-group -lstdc++ -lsupc++ -Wl,--end-group")
```

Redirect the UART – console.h

C console.h ×

Core > Inc > C console.h > ...

```
1  #ifndef INC_CONSOLE_H_
2  #define INC_CONSOLE_H_
3
4  #include "usart.h"
5
6  #ifdef __cplusplus
7  extern "C" {
8  #endif
9
10 // Serial port to use.
11 void SetUart(UART_HandleTypeDef* huart);
12
13 #ifdef __cplusplus
14 }
15 #endif
16
17 #endif /* INC_CONSOLE_H_ */
```

Redirect the UART – console.c

C console.c ×

Core > Src > C console.c > ...

```c
#include <stdio.h>
#include "usart.h"
#include "console.h"

static UART_HandleTypeDef* huart;

void SetUart(UART_HandleTypeDef* huart_)
{
 huart=huart_;
 setbuf(stdout, NULL);
}

#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif

PUTCHAR_PROTOTYPE
{
 if(huart)
  HAL_UART_Transmit(huart, (uint8_t*)&ch, 1, HAL_MAX_DELAY);
 return ch;
}
```

Redirect the UART – Modify cmake/stm32cubemx/CMakeLists.txt

CMakeLists.txt 1

cmake > stm32cubemx > CMakeLists.txt

```
19
20    # STM32CubeMX generated application sources
21    set(MX_Application_Src
22        ${CMAKE_SOURCE_DIR}/Core/Src/main.cpp
23        ${CMAKE_SOURCE_DIR}/Core/Src/console.c
24        ${CMAKE_SOURCE_DIR}/Core/Src/gpio.c
25        ${CMAKE_SOURCE_DIR}/Core/Src/usart.c
26        ${CMAKE_SOURCE_DIR}/Core/Src/stm32g0xx_it.c
27        ${CMAKE_SOURCE_DIR}/Core/Src/stm32g0xx_hal_msp.c
28        ${CMAKE_SOURCE_DIR}/Core/Src/sysmem.c
29        ${CMAKE_SOURCE_DIR}/Core/Src/syscalls.c
30        ${CMAKE_SOURCE_DIR}/startup_stm32g031xx.s
31    )
```

Test It!

```
128   SetUart(&huart2); // Select UART2
129
130   printf("Using UART2\r\n");
131   printf("pi = %3.2f\r\n",numbr);
132   while(1);
133
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR

+ Open an additional monitor

Monitor Mode: Serial | View Mode: Text | Port: /dev/ttyACM0 - STMicroelectronics | Baud rate: 115200

```
Using UART2
pi = 3.14
```

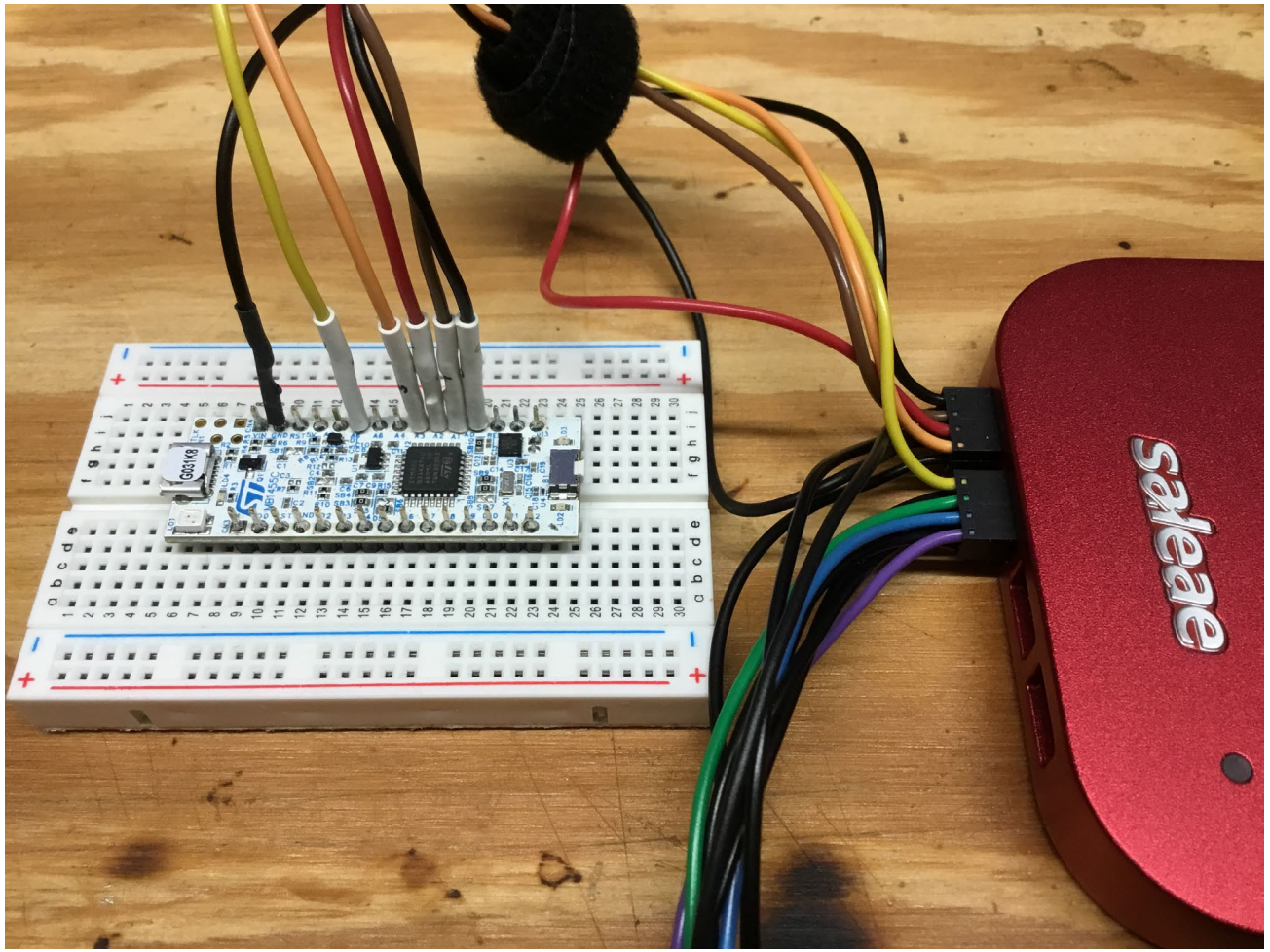
Next Time…

Thank you for attending!!!

Please consider the resources below:

- **Today's Download Package**

MORE TO COME..



CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets