CEC Continuing Education Center

DesignNews

PIC Microcontroller Embedded Development Using the CCS PIC MCU C Compiler

Day 4:
The CCS Long Range RF Kit

Sponsored by

Digi-Key ELECTRONICS

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

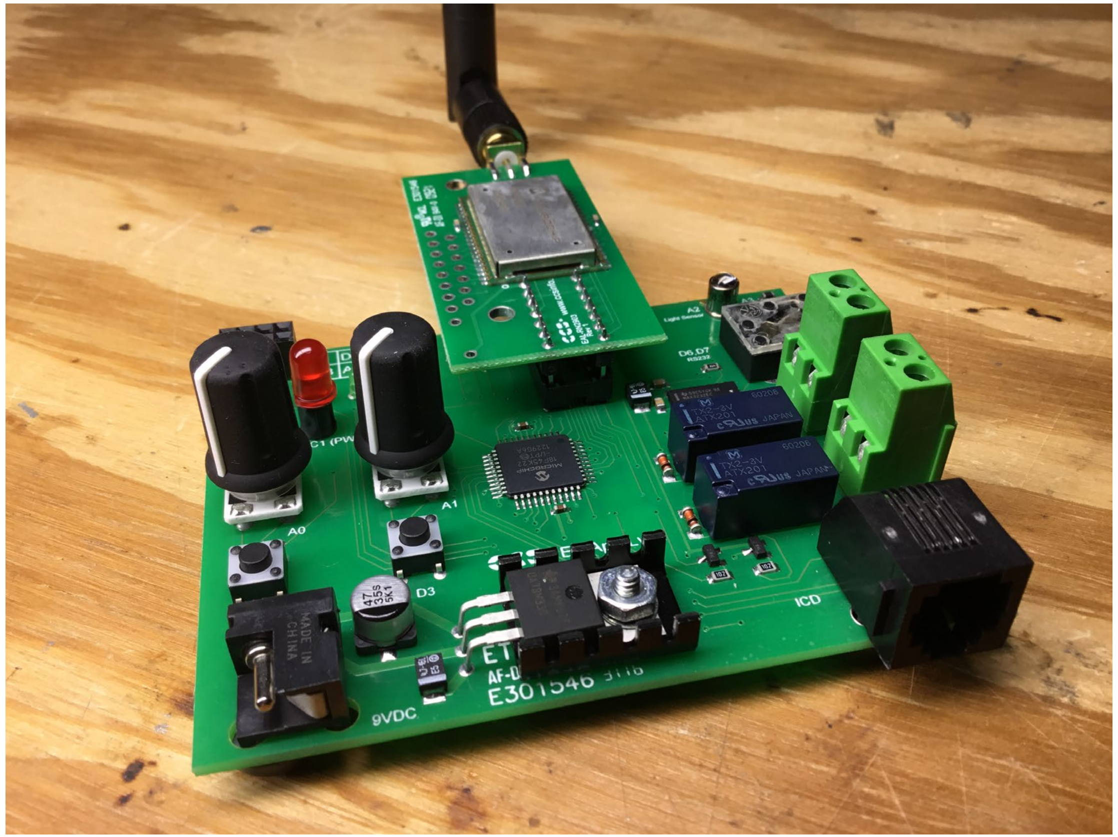
Visit 'Lecturer Profile' in your console for more details.

CEC Continuing Education Center

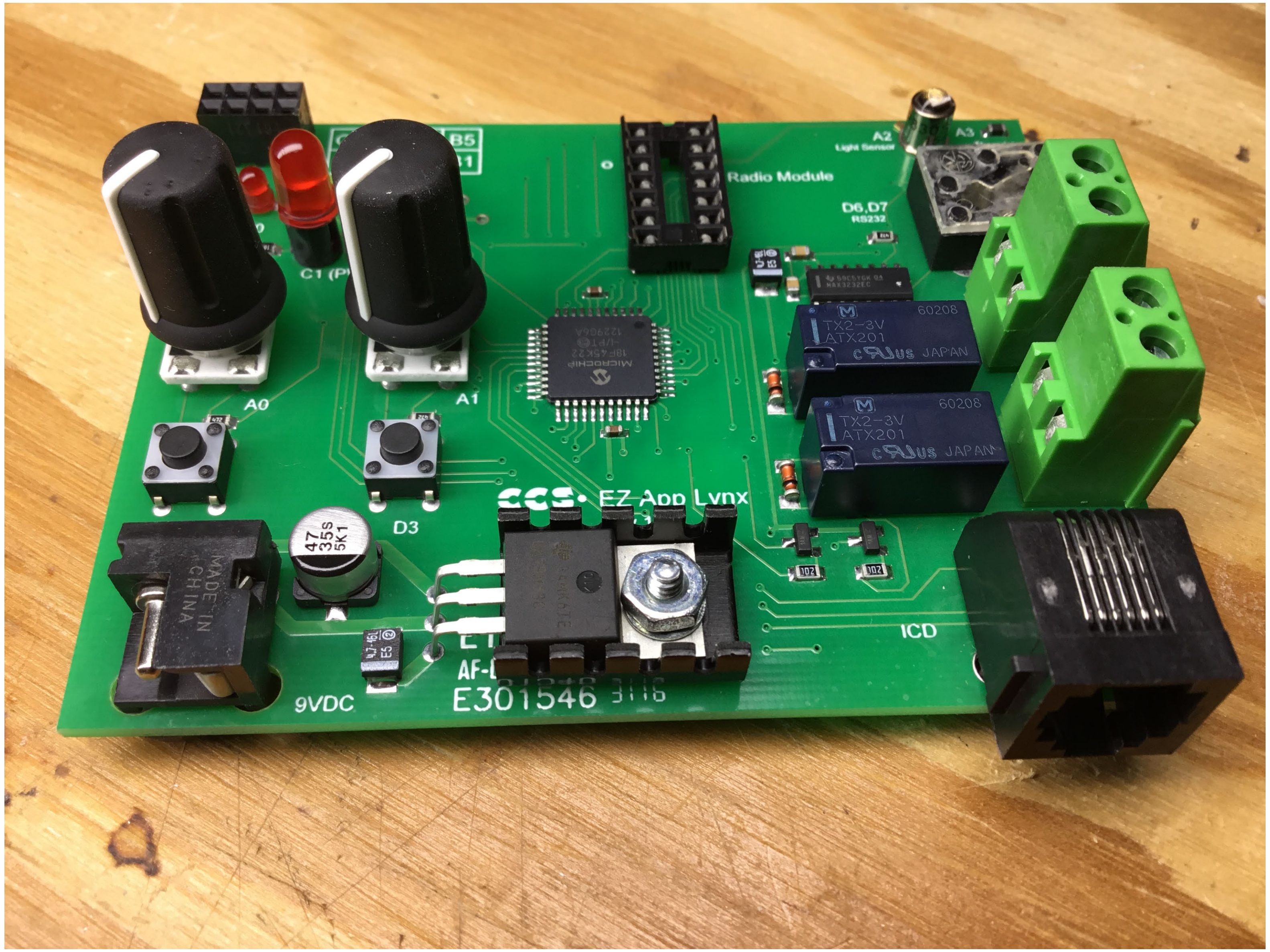
Sponsored By Digi-Key Corporation

AGENDA

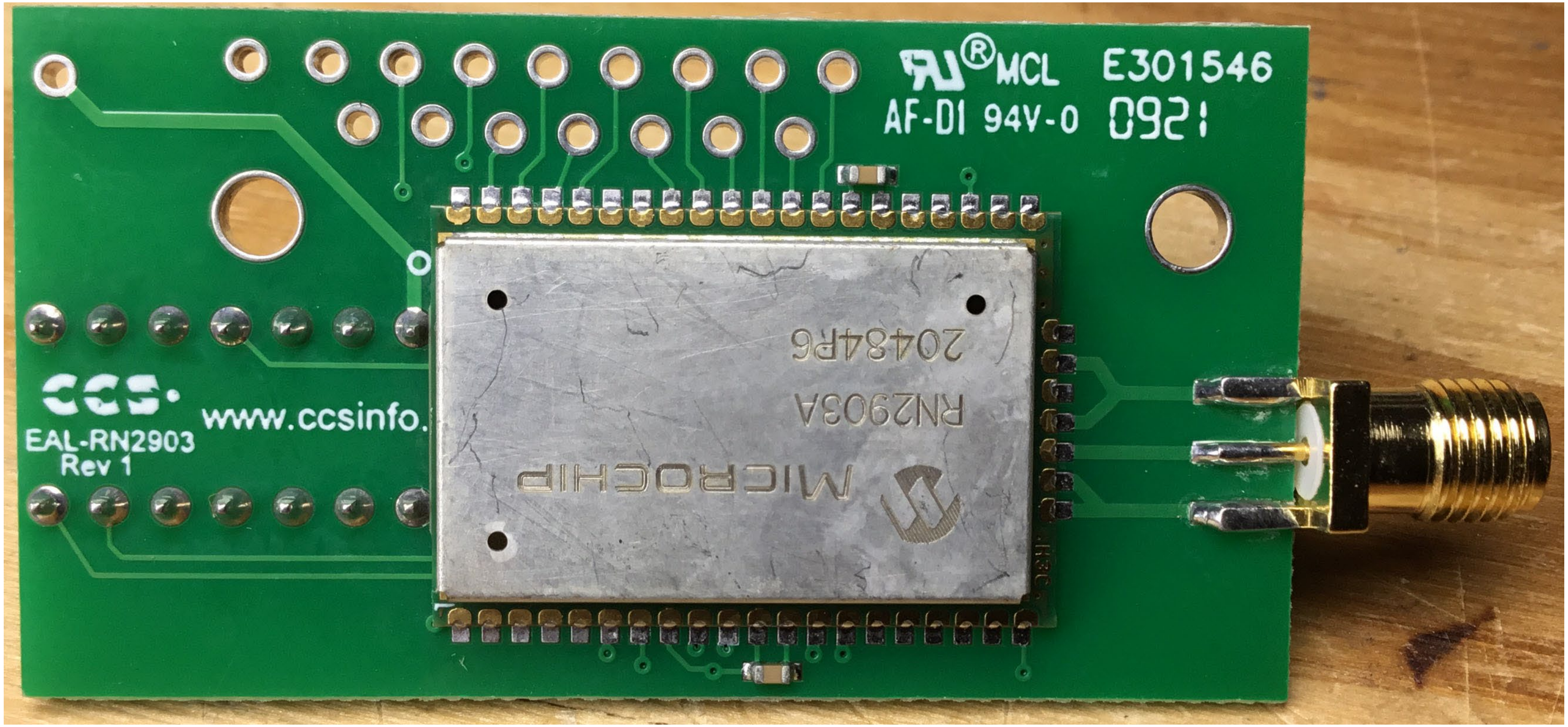
- **Long Range Kit Hardware**
- **devkit_lora.h**
- **Communicate with the RN2903**
- **Code an RN2903 Application**



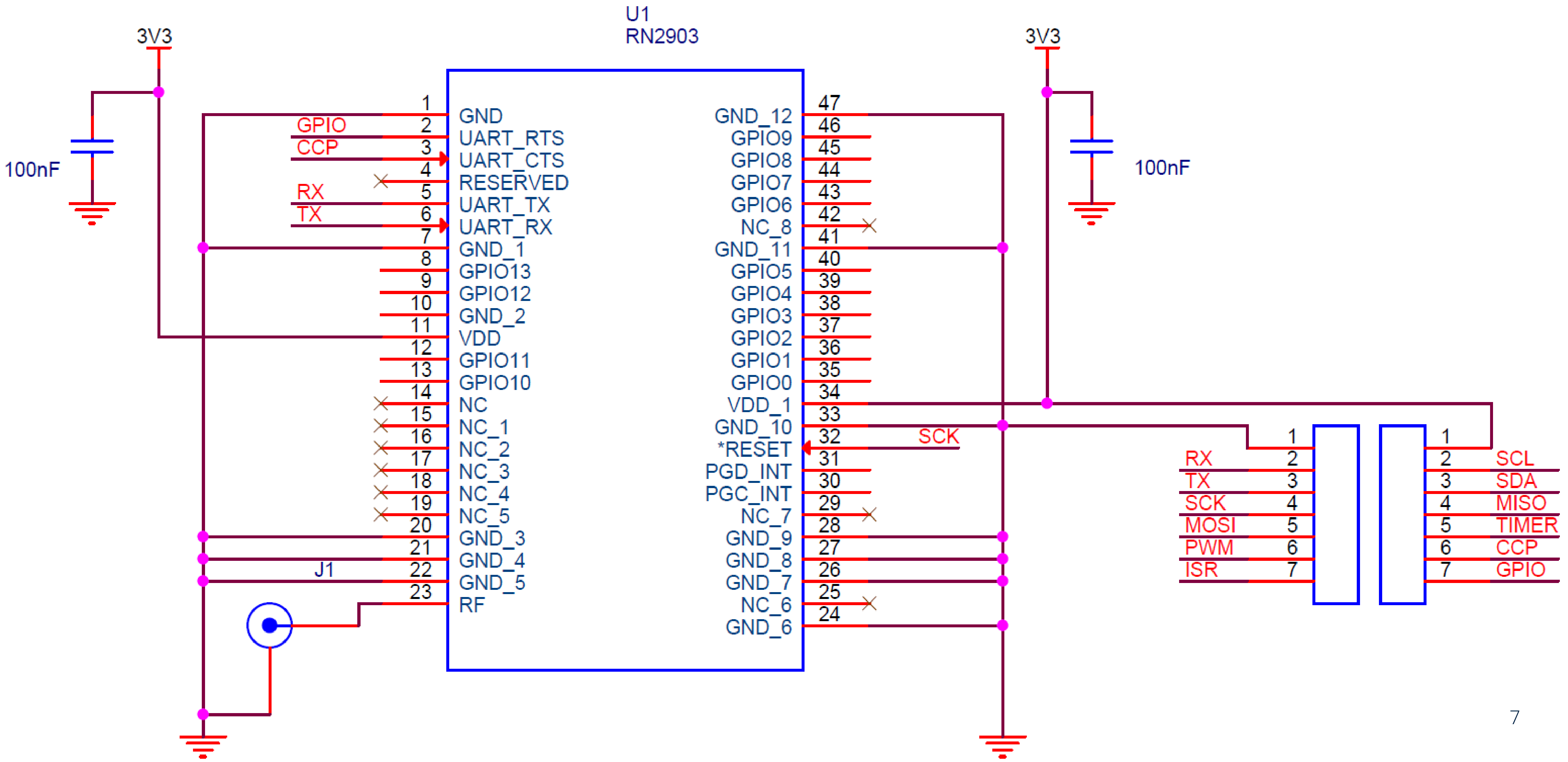
PIC18F45K22 Hardware



RN2903 Hardware



RN2903 Hardware



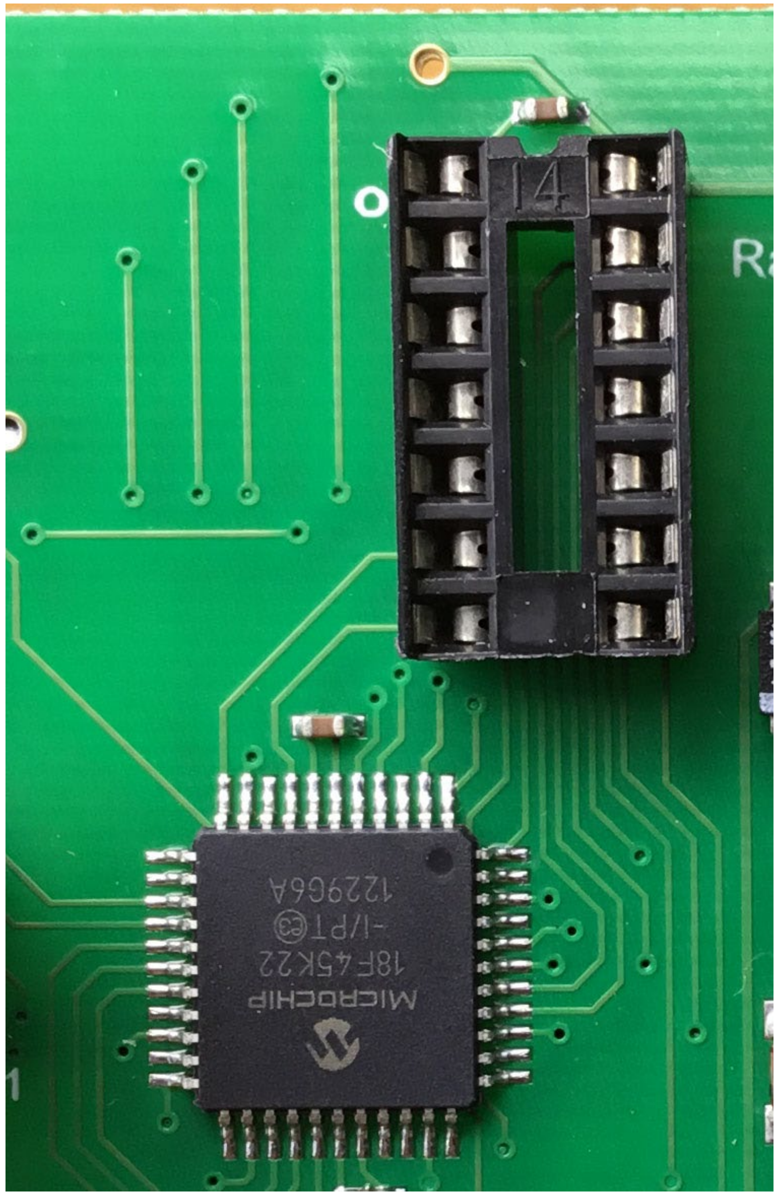
Sponsored By

PIC18F45K22 Hardware

```
#include <18F45K22.h>

  #if defined(DEVICE_USE_ICD)
     #device ICD=TRUE
  #endif

  #device ADC=10

  // Datasheet shows 2% error at 16mhz, any other values aren't documented.
  #use delay(internal=16MHz)

  // NOTE: bootloader cannot change the fuses/config-bits!!
  #fuses NOWDT      //WDT is not permanently enabled.  setup_wdt() can be used to turn it on
  #fuses WDT1024    //WDT perdiod (no prescalar) ~4.1ms.  1024*4.1ms = ~4s period
  #fuses NOWRT      //program memory not write protected
  #fuses WRTB       //write protect boot-block
  #fuses WRTC       //write protect config bits
  #fuses NOWRTD     //data EEPROM not write protected
```

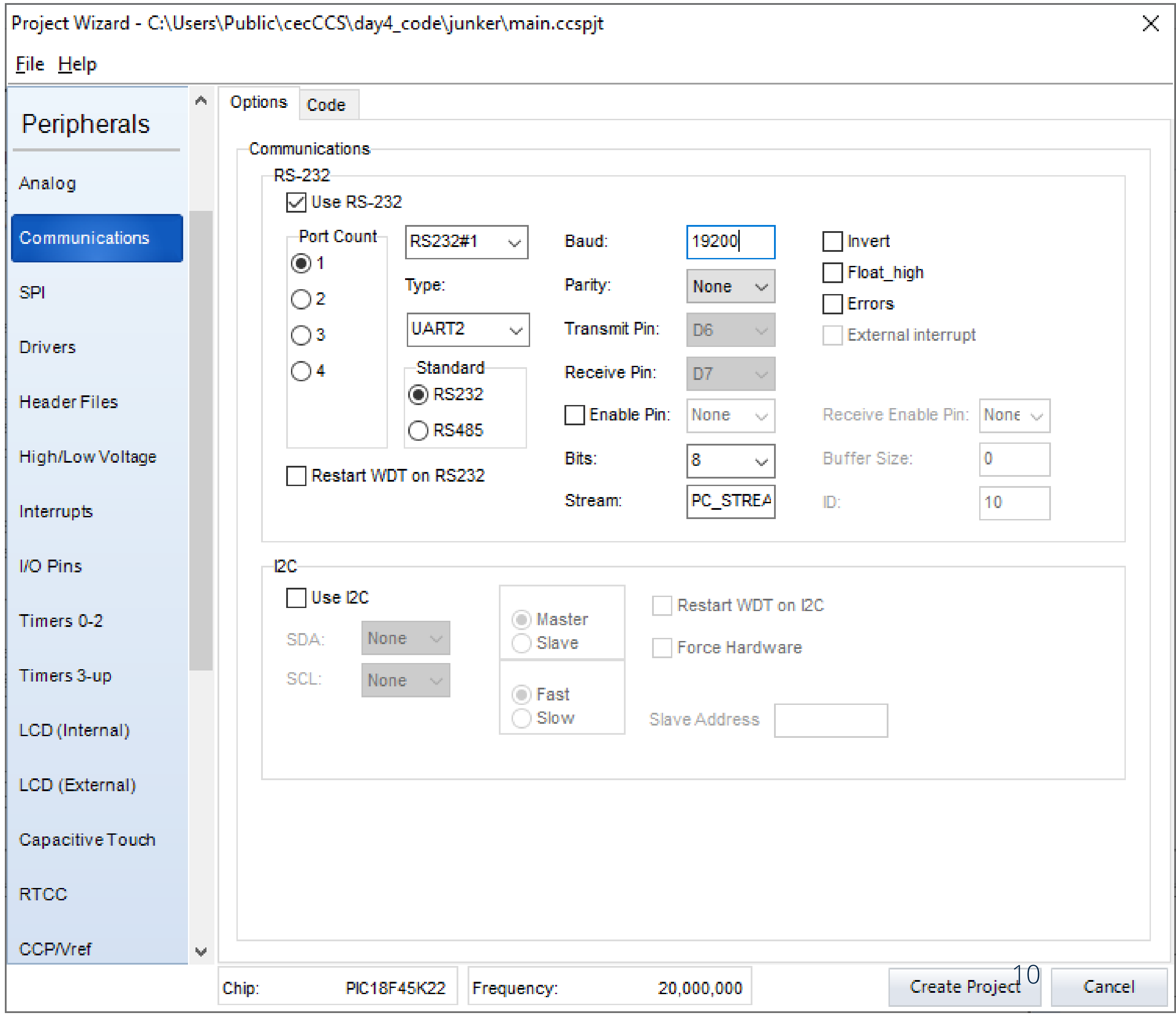
PIC18F45K22 Hardware

```
// UART1 connected to PDIP daughterboard socket.
  #define RN2903_TX_PIN                   PIN_C6
  #define RN2903_RX_PIN                   PIN_C7
  //#define RN2903_SERIAL_BUFFER_SIZE   0

  #ifdef NO_SERIAL_PORT
     #define user_printf(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p)
     #define user_putc(c)
     #define user_getc()
     #define user_kbhit()
  #else
     #ifndef USE_ICD_FOR_SERIAL
        // UART2 is connected to an RS232 level converter.
        #use rs232(UART2, baud=19200, stream=PC_STREAM)
     #else
        #use rs232(ICD, baud=19200, stream=PC_STREAM, NODELAY)
     #endif

     #define user_printf     printf

     void user_putc(char c)
     {
        fputc(c, PC_STREAM);
     }

     #define user_getc()     fgetc(PC_STREAM)
     #define user_kbhit()    kbhit(PC_STREAM)
 #endif
```



PIC18F45K22 Hardware

```
// UART1 connected to PDIP daughterboard socket.
  #define RN2903_TX_PIN                  PIN_C6
  #define RN2903_RX_PIN                  PIN_C7
  //#define RN2903_SERIAL_BUFFER_SIZE   0

  #ifdef NO_SERIAL_PORT
     #define user_printf(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p)
     #define user_putc(c)
     #define user_getc()
     #define user_kbhit()
  #else
     #ifndef USE_ICD_FOR_SERIAL
        // UART2 is connected to an RS232 level converter.
        #use rs232(UART2, baud=19200, stream=PC_STREAM)
     #else
        #use rs232(ICD, baud=19200, stream=PC_STREAM, NODELAY)
     #endif

     #define user_printf      printf

     void user_putc(char c)
     {
        fputc(c, PC_STREAM);
     }

     #define user_getc()     fgetc(PC_STREAM)
     #define user_kbhit()    kbhit(PC_STREAM)
 #endif
```



Communications

RS-232

☑ Use RS-232

Port Count: (● 1) ○ 2 ○ 3 ○ 4

RS232#1	Baud:	19200	☐ Invert		
Type:	Parity:	None	☐ Float_high		
UART2	Transmit Pin:	D6	☐ Errors		
Standard: (● RS232) ○ RS485	Receive Pin:	D7	☐ External interrupt		
	☐ Enable Pin:	None	Receive Enable Pin:	None	
	Bits:	8	Buffer Size:	0	
☐ Restart WDT on RS232	Stream:	PC_STREA	ID:	10	

I2C

☐ Use I2C

SDA: None
SCL: None

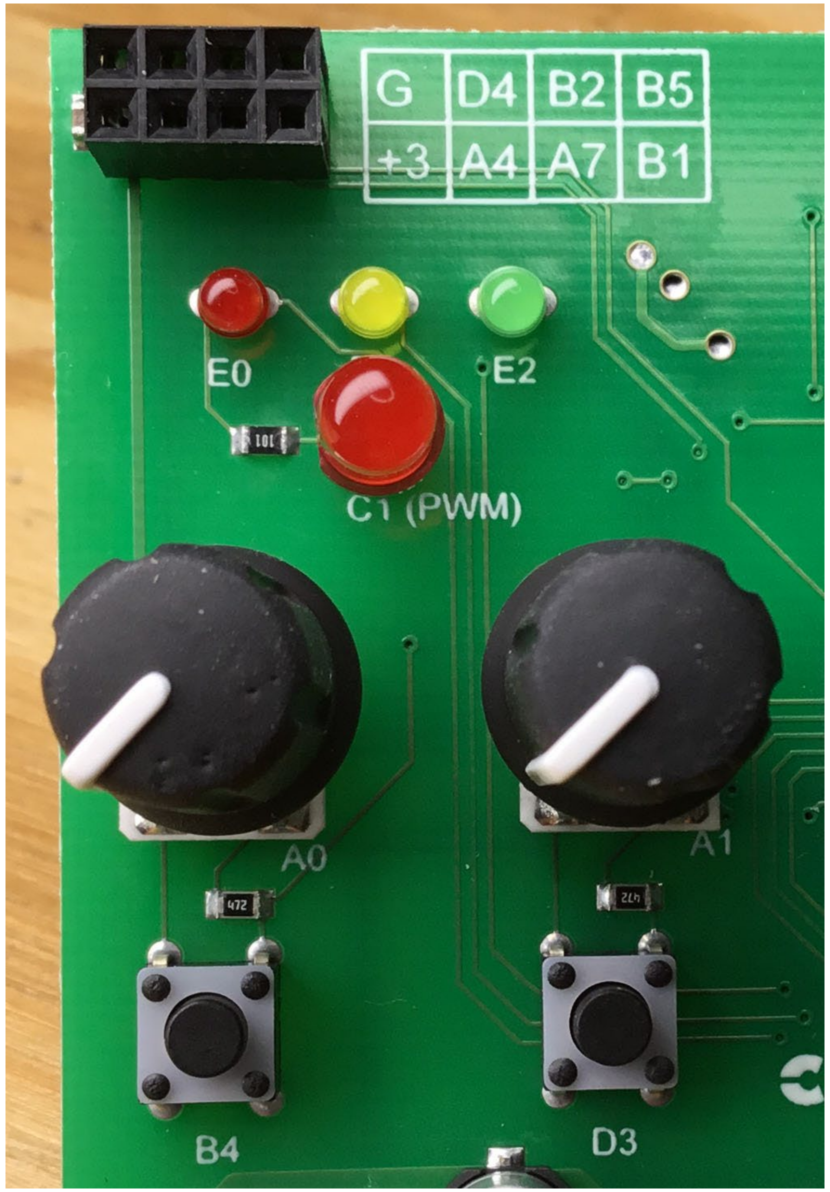
(● Master) ○ Slave
(● Fast) ○ Slow

☐ Restart WDT on I2C
☐ Force Hardware
Slave Address

Chip: PIC18F45K22
Frequency: 20,000,000

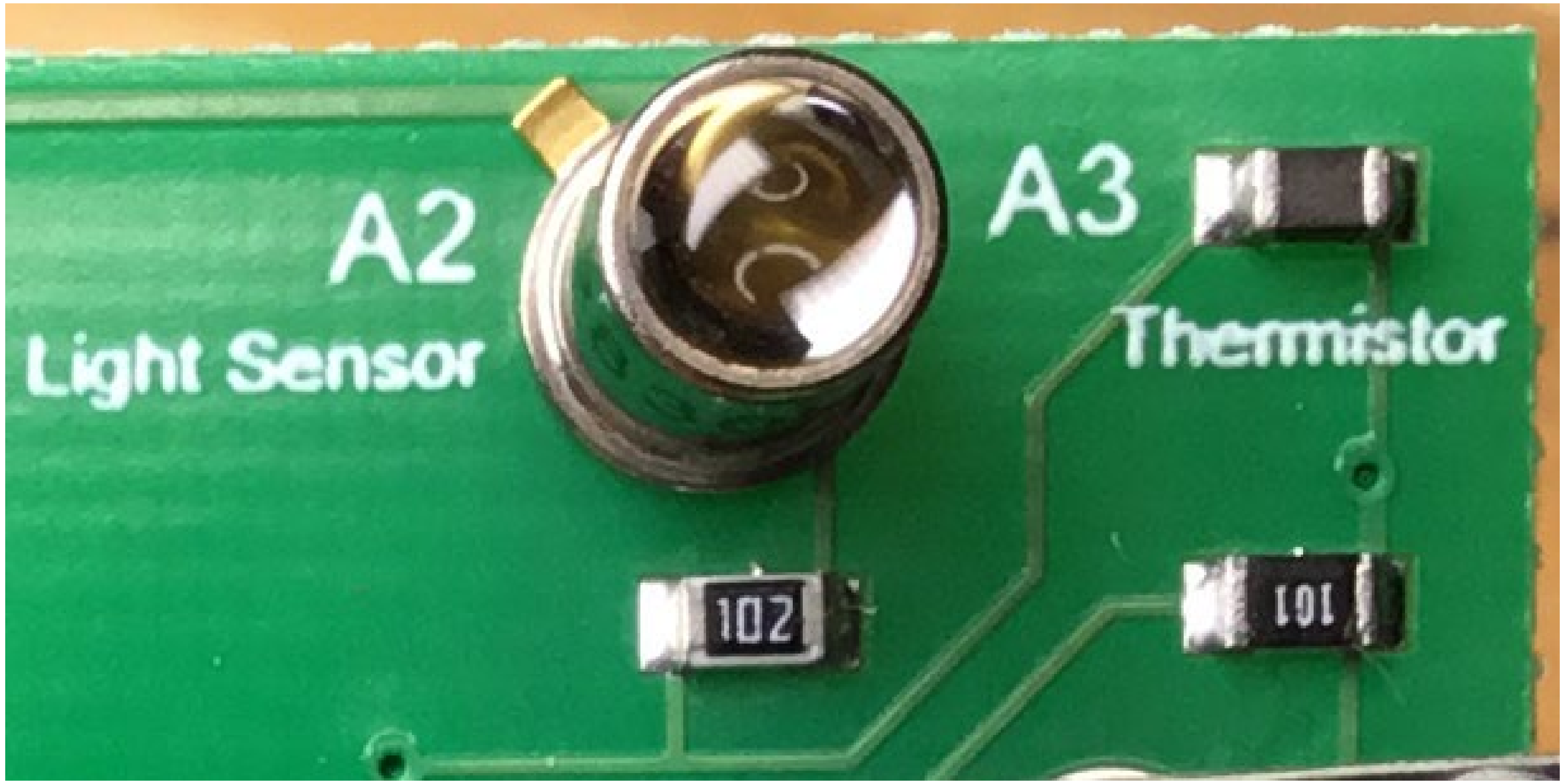
PIC18F45K22 Hardware

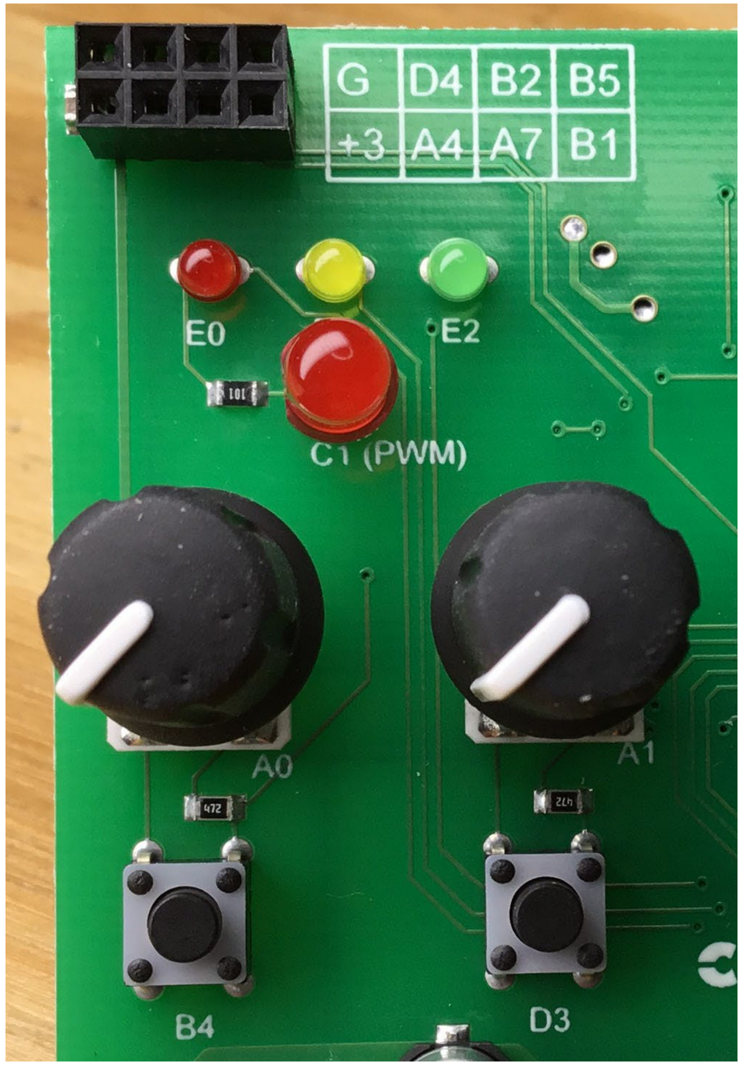
```
// Small LEDs.
  // set pin high to turn on LED, set pin low to turn off LED.
  #define PIN_LED_RED     PIN_E0
  #define PIN_LED_YELLOW  PIN_E1
  #define PIN_LED_GREEN   PIN_E2

  // BIG LED.
  #define PIN_LED_BIG_RED    PIN_C1

  // push-buttons.
  // the pin will be read low when button is held pressed.
  // the pin will be read high when button is idle (not pressed).
  #define PIN_INPUT_BUTTON0  PIN_D3
  #define PIN_INPUT_BUTTON1  PIN_B4

  #define BUTTON_PRESSED        0
  #define BUTTON_NOT_PRESSED    1
```



PIC18F45K22 Hardware

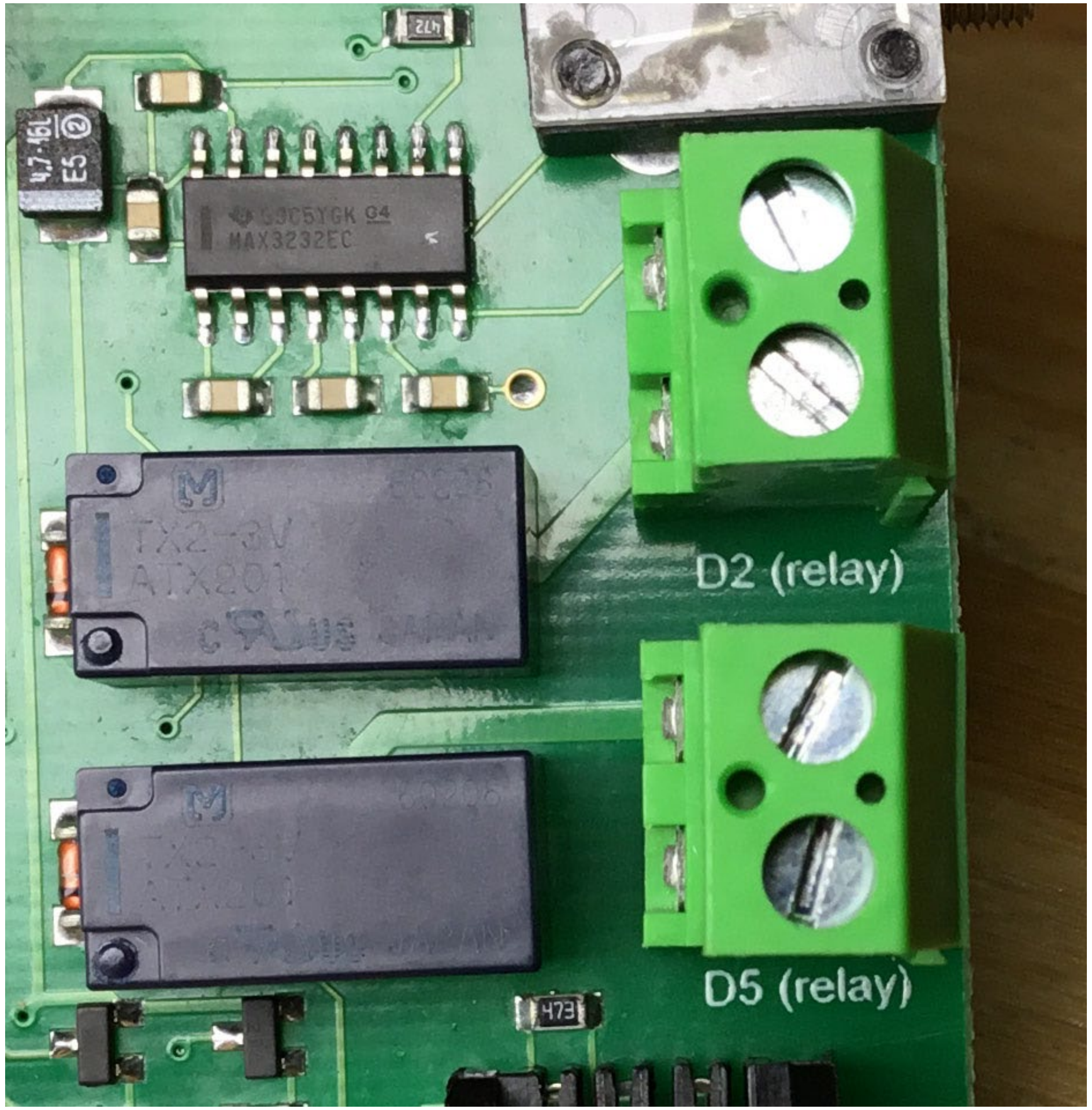
```
// analog channels.
  // you can pass these to read_adc.
  // don't forget setup_adc_ports(sAN0 | sAN1 | sAN2 | sAN3) and setup_adc()
  #define AN_CHANNEL_POT0           0  //PIN_A0
  #define AN_CHANNEL_POT1           1  //PIN_A1
  #define AN_CHANNEL_PHOTO          2  //PIN_A2 - photo diode
  #define AN_CHANNEL_THERMISTOR     3  //PIN_A3
```





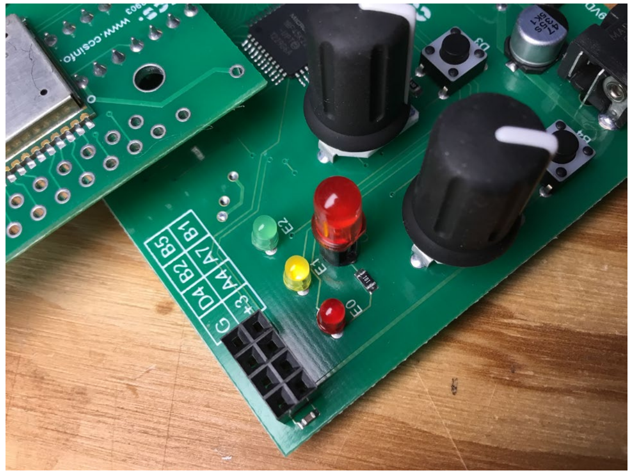
PIC18F45K22 Hardware

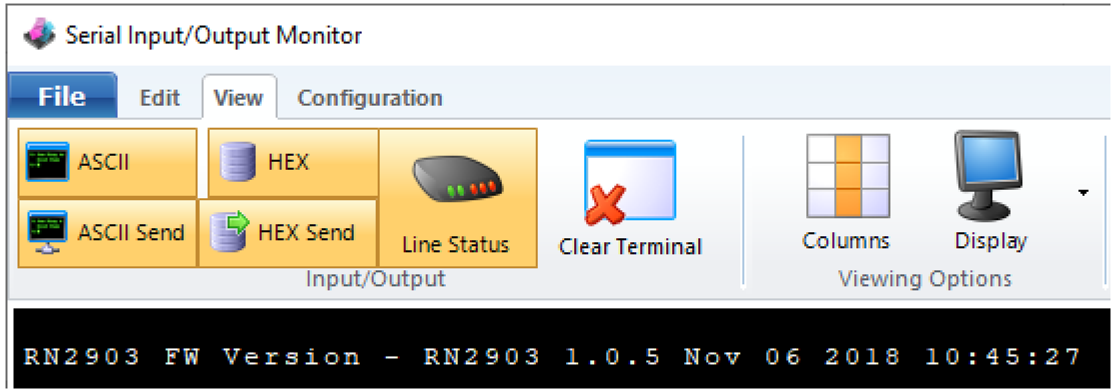
```
// Relays.
  #define PIN_OUTPUT_RELAY0   PIN_D2
  #define PIN_OUTPUT_RELAY1   PIN_D5
```



Get the RN2903 Firmware Version

```
/*
   Optional if defined it uses the ICD for the PC serial communication for this
   example.  If this is defined, then the program should be programmed with the
   ICD because after program it auto launches the Serial Input/Output Monitor
   program setup correctly for communicating through the ICD.
*/
#define  USE_ICD_FOR_SERIAL

#include <devkit_lora.h>
#include <rn2903.c>

void main(void)
{
   char vStr[50];

   RN2903SerialInit();  //initialize the RN2903 serial interface

   RN2903Version(vStr); //read the RN2903 modules FW version

   if(vStr[0] != '\0')
   {
      user_printf(user_putc, "\r\nRN2903 FW Version - %s\r\n", vStr);
      output_high(PIN_LED_YELLOW);   //turn Yellow LED on indicating that version was read from RN2903 module

   }
   else
      output_high(PIN_LED_RED);     //turn Red LED on indicating that version wasn't read from RN2903 module

   while(TRUE);
}
```

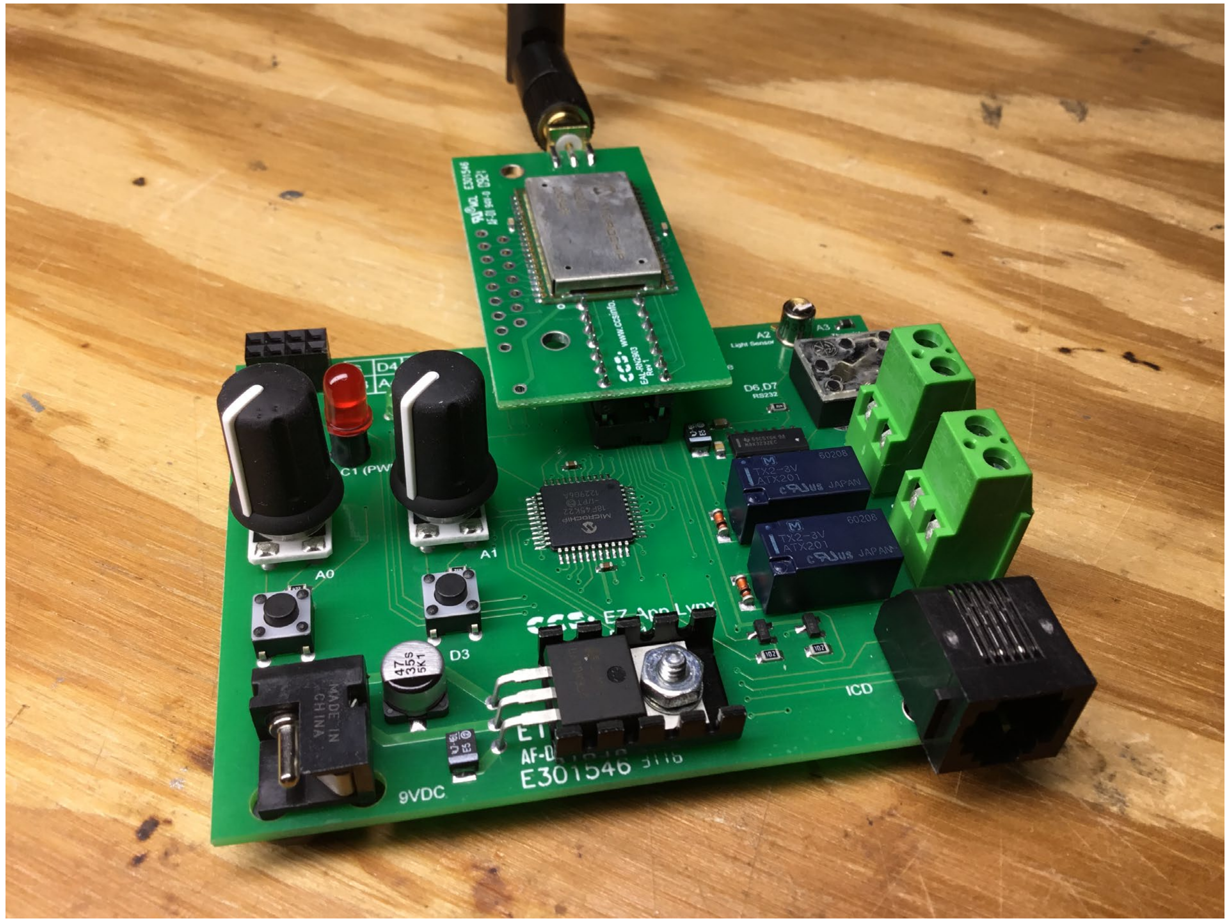




```
RN2903 FW Version - RN2903 1.0.5 Nov 06 2018 10:45:27
```

End Station Code

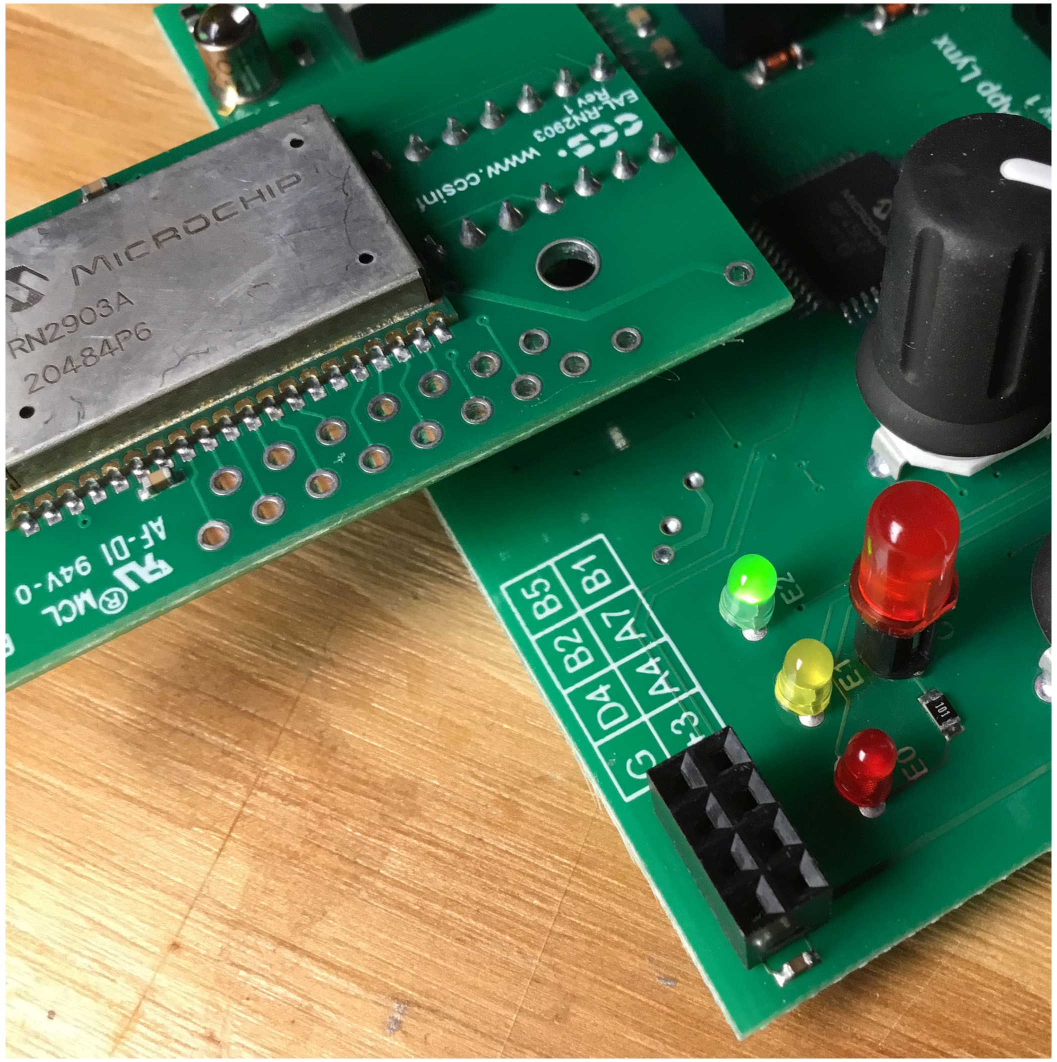
```
#include <devkit_lora.h>

// Include a tick timer driver and make appropriate defines, the lora.c driver
// requires a tick timer.
#include <tick.c>
#define TickDifference(a,b)      (a-b)

#define _lora_tick_t                    TICK
#define _LORA_TICKS_PER_SECOND          TICKS_PER_SECOND
#define _lora_tick_difference(a,b)      TickDifference(a,b)
#define _lora_tick_get()                TickGet()

// Set device type to an End-Device
#define LORA_DEVICE_TYPE    LORA_DEVICE_TYPE_ED

// Include the LoRa P2P driver's header file, needs to be included before
// rn2903.c so that the required LoRa P2P driver functions in rn2903.c are
// accessible.
#include <lora.h>

// Include the RN2903 module driver and the LoRa P2P driver
#include <rn2903.c>
#include <lora.c>

#define ACK     1
#define NACK    0

void InitHW(void)
{
   setup_adc_ports(sAN0, VSS_VDD);
   setup_adc(ADC_CLOCK_INTERNAL);
   set_adc_channel(AN_CHANNEL_POT0);
}
```



Sponsored By

End Station Code

```
void main(void)
{
   int1 Initialized;
   int1 cLevel, pLevel;
   uint16_t Reading;
   int1 Sent = FALSE;
   lora_rx_message_t Message;

   //Initialize the LoRa P2P driver, this always need called before any other
   //LoRa P2P functions.  lora_init() also makes necessary calls to initialize
   //the RN2903 module.
   Initialized = lora_init();

   if(Initialized)
   {
      output_high(PIN_LED_GREEN);

      InitHW();

      pLevel = input(PIN_INPUT_BUTTON1);
   }
   else
      output_high(PIN_LED_RED);
```



Sponsored By

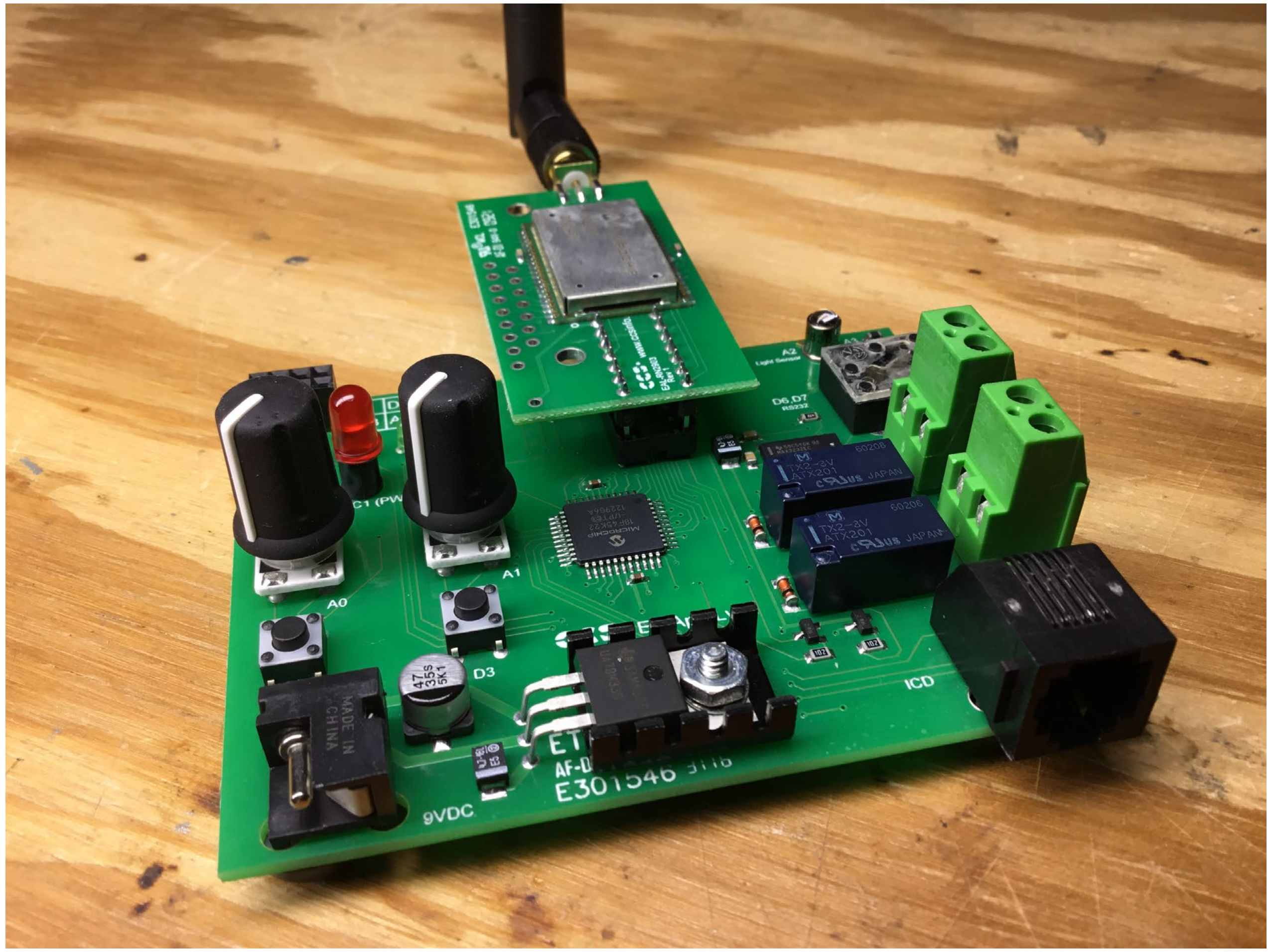
End Station Code

```
if(Sent == FALSE)
        {
           cLevel = input(PIN_INPUT_BUTTON1);

           if(cLevel != pLevel)
           {
              delay_us(50);

              if(input(PIN_INPUT_BUTTON1) != pLevel)
              {
                 if(cLevel == BUTTON_PRESSED)
                 {
                    output_low(PIN_LED_RED);

                    Reading = read_adc();   //Read current ADC reading of pot A0.

                    //Send ADC reading to base station.
                    if(lora_put_message(LORA_BS_DEVICE_ADDR, (uint8_t *)&Reading, 2))
                    {
                       Sent = TRUE;

                       output_high(PIN_LED_YELLOW);
                    }
                    else
                       output_high(PIN_LED_RED);
                 }

                 pLevel = cLevel;
              }
           }
        }
```

Sponsored By

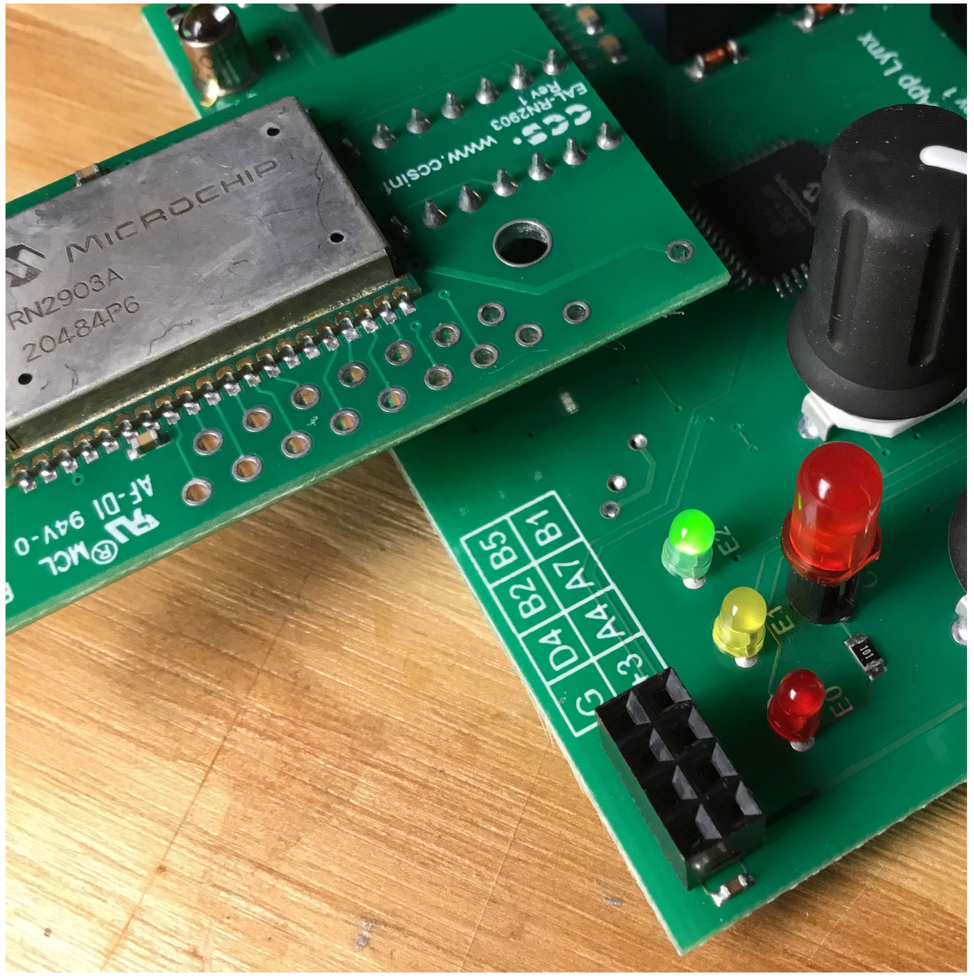
Base Station Code

```
 Optional if defined it uses the ICD for the PC serial communication for this
   example.  If this is defined, then the program should be programmed with the
   ICD because after program it auto launches the Serial Input/Output Monitor
   program setup correctly for communicating through the ICD.
*/
#define USE_ICD_FOR_SERIAL

#include <devkit_lora.h>

// Include a tick timer driver and make appropriate defines, the lora.c driver
// requires a tick timer.
#include <tick.c>
#define TickDifference(a,b)      (a-b)
#define _lora_tick_t                TICK
#define _LORA_TICKS_PER_SECOND      TICKS_PER_SECOND
#define _lora_tick_difference(a,b)  TickDifference(a,b)
#define _lora_tick_get()            TickGet()

// Set device type to a Base Station
#define LORA_DEVICE_TYPE   LORA_DEVICE_TYPE_BS

// Include the LoRa P2P driver's header file, needs to be included before
// rn2903.c so that the required LoRa P2P driver functions in rn2903.c are
// accessible.
#include <lora.h>

// Include the RN2903 module driver and the LoRa P2P driver
#include <rn2903.c>
#include <lora.c>

#define ACK     1
#define NACK    0

// Setup PWM on Big Red LED, CCP2, to output ~1 kHz frequency with 10 bit
// resolution, initialize duty to 0%.
#use pwm(output=PIN_LED_BIG_RED, frequency=1kHz, bits=10, duty=0)
```



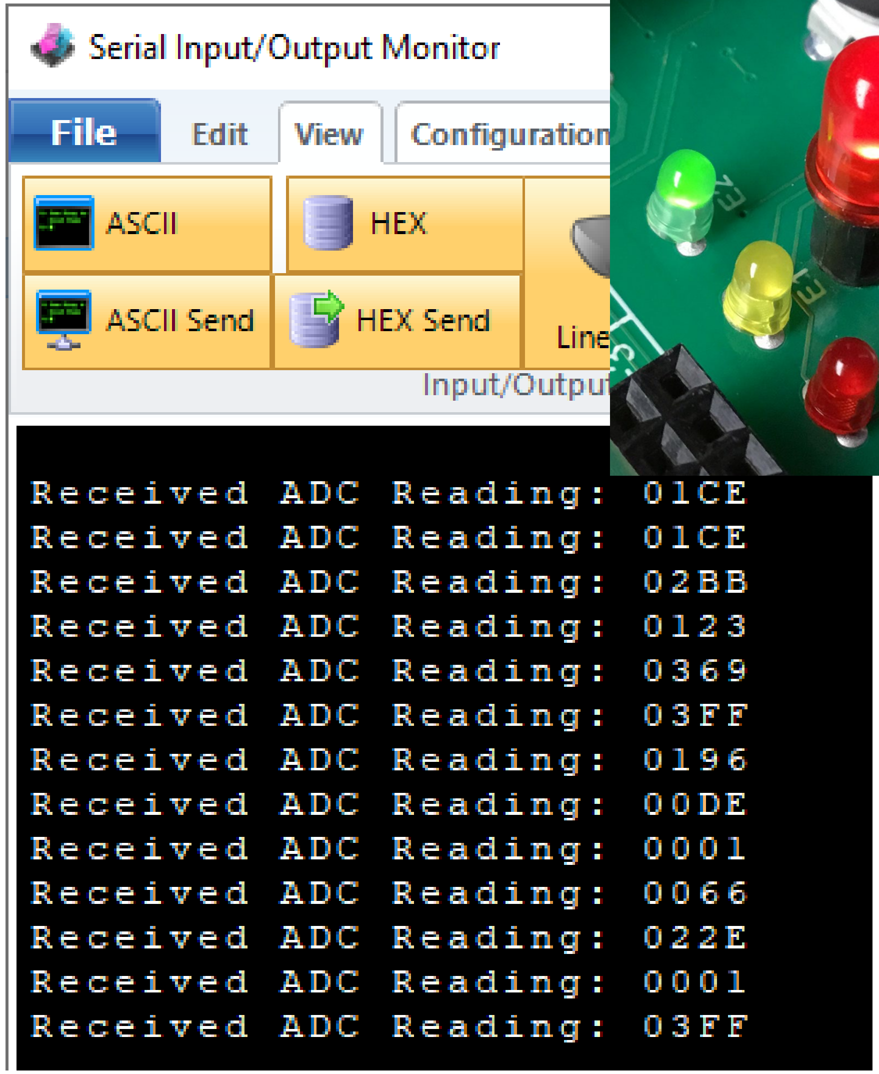
Base Station Code

```c
void main(void)
{
   lora_rx_message_t Message;
   uint8_t txData[1] = {ACK};
   uint16_t Reading;
   int1 Initilized;

   //Initialize the LoRa P2P driver, this always need called before any other
   //LoRa P2P functions.  lora_init() also makes necessary calls to initialize
   //the RN2903 module.
   Initilized = lora_init();

   if(Initilized)
      output_high(PIN_LED_GREEN);
   else
   {
      output_high(PIN_LED_RED);

      user_printf(user_putc, "\r\nRN2903 Module failed to initialize, try power cycling board.");
   }
```



Base Station Code

```c
while(TRUE)
  {
     if(Initilized)
     {
        //LoRa P2P task function to maintain state machine for sending and
        //receiving messages.
        lora_task();

        //Check if a new message has been received, since this device is setup
        //as a base station it is always listen for a message.  The only
        //exception to this after it receive a message meant for it.  In that
        //case it will wait for LORA_RX_TIME milliseconds for a message to
        //send to be loaded into it's TX buffer, during that time it's not
        //listening for message.
        if(lora_has_message())
        {
           //Retrieve message from software buffer.
           lora_get_message(&Message);

           if(Message.Length == 2)
           {
              Reading = make16(Message.Data[1], Message.Data[0]);

              user_printf(user_putc, "\r\nReceived ADC Reading: %04LX", Reading);

              //Set PWM duty to the ADC reading received from end-device.
              pwm_set_duty(Reading);
           }

           //Load ACK into TX buffer to send as a response.
           lora_put_message(Message.Address, txData, 1);
```



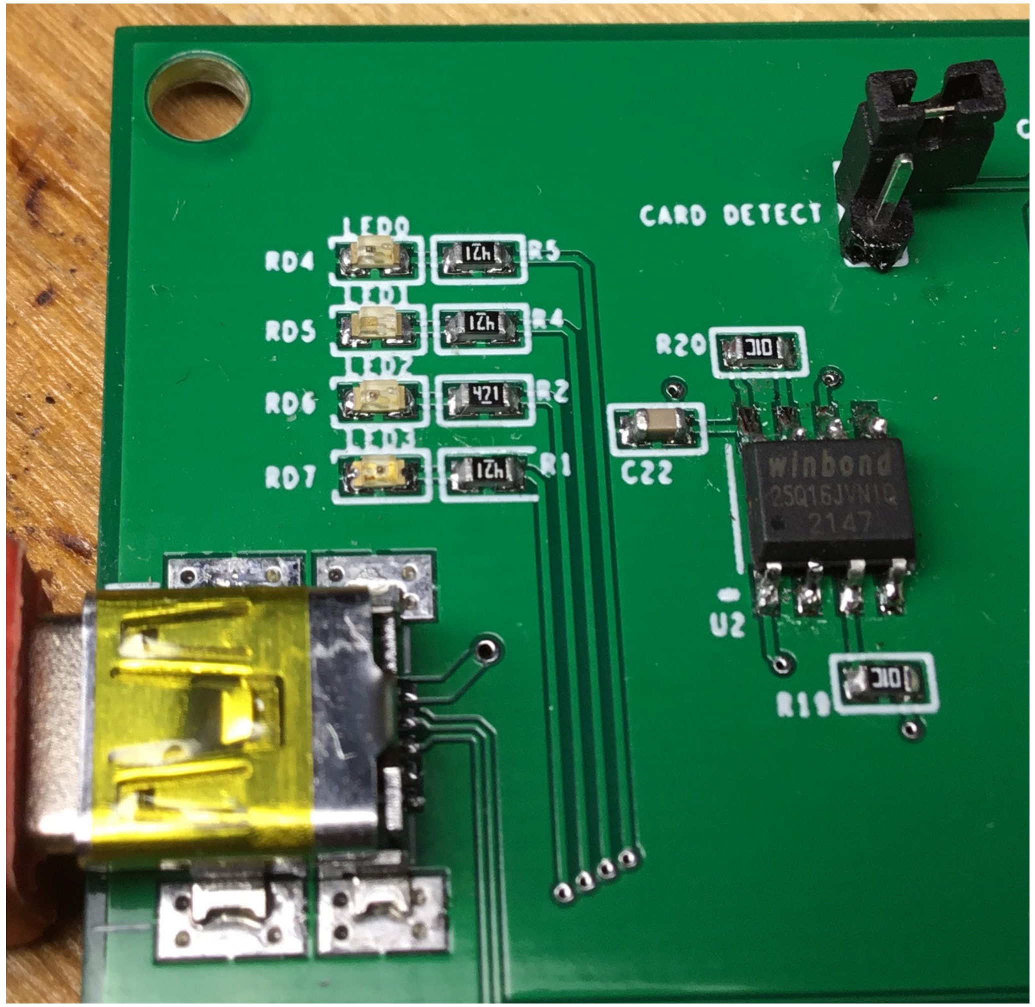
```
Received ADC Reading: 01CE
Received ADC Reading: 01CE
Received ADC Reading: 02BB
Received ADC Reading: 0123
Received ADC Reading: 0369
Received ADC Reading: 03FF
Received ADC Reading: 0196
Received ADC Reading: 00DE
Received ADC Reading: 0001
Received ADC Reading: 0066
Received ADC Reading: 022E
Received ADC Reading: 0001
Received ADC Reading: 03FF
```

End Station Code

```
while(TRUE)
  {
     if(Initialized)
     {
        //LoRa P2P task function to maintain state machine for sending and
        //receiving messages.
        lora_task();

        //Check if a new message has been received, since this device is setup
        //as an End-Device it can only receive a messages after sending a
        //message.  After sending a message it will listen for LORA_RX_TIME
        //milliseconds.  The time can be adjusted by defining LORA_RX_TIME
        //before the lora.c driver file is include, the default time is 2000
        //milliseconds.
        if(lora_has_message())
        {
           //Retrieve message from software buffer.
           lora_get_message(&Message);

           if(Message.Address == LORA_BS_DEVICE_ADDR)
           {
              if(Sent)
              {
                 Sent = FALSE;

                 output_low(PIN_LED_YELLOW);

                 if(Message.Data[0] == NACK)
                    output_high(PIN_LED_RED);
              }
           }
        }
```

MORE TO COME..

Thank you for attending!!!



Please consider the resources below:

- ccsinfo.com
- CCS C Compiler Manual
- Master and Command C for PIC MCU (PDF)

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets