CEC Continuing Education Center

DesignNews

Developing IoT Applications with Nordic nRF Modules

Day 2:

BLE Application Development Using Raytac Modules

Sponsored by

DigiKey

informa markets

Webinar Logistics

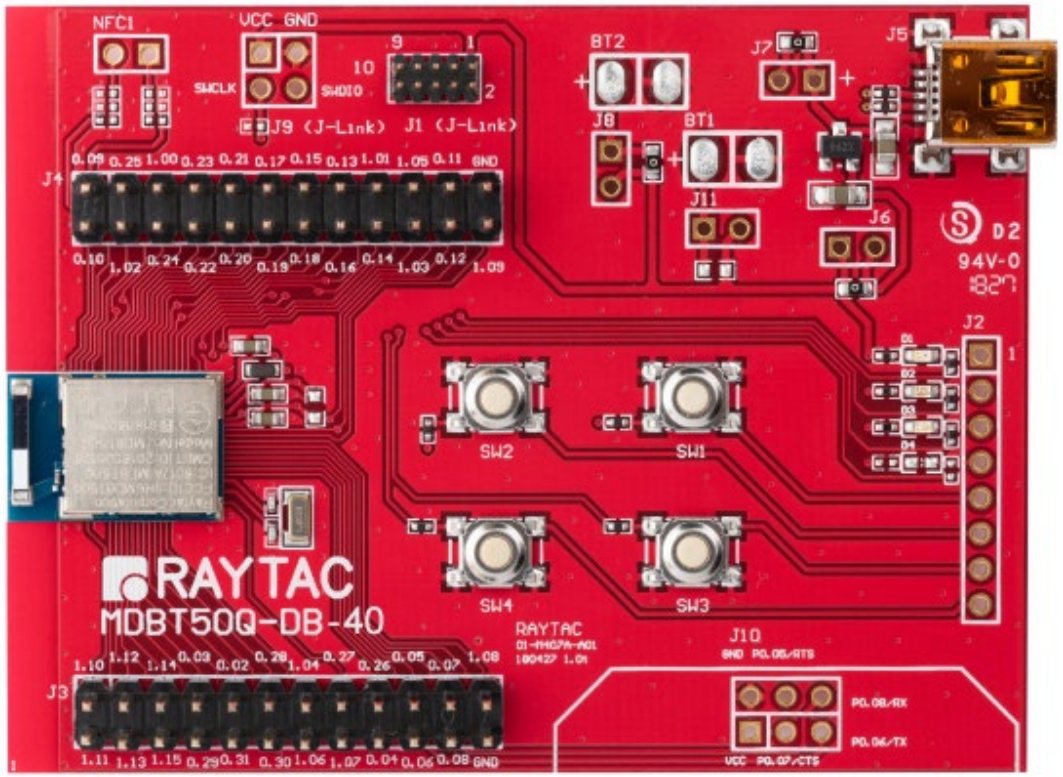
- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Fred Eady

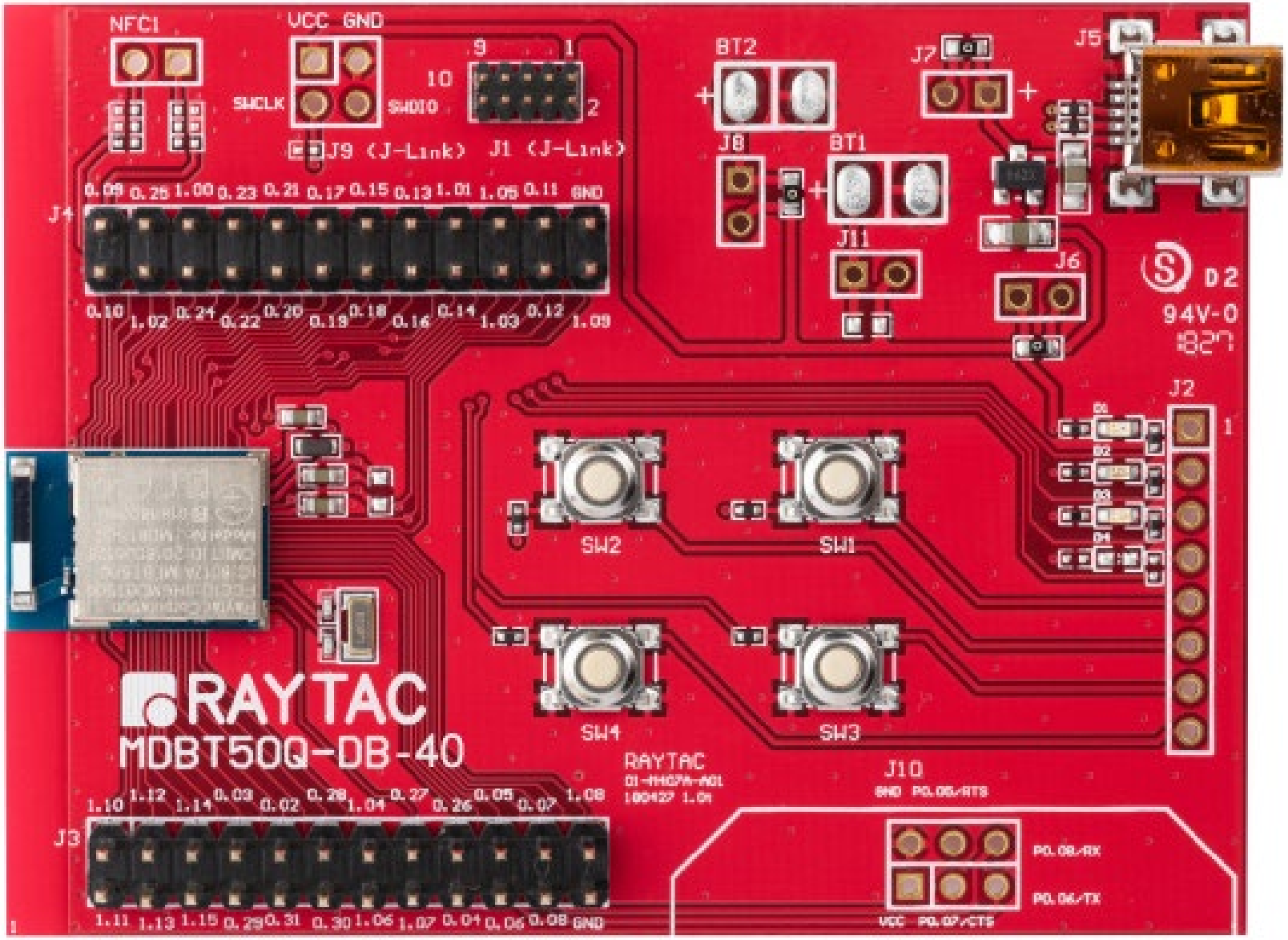
Visit ‘Lecturer Profile’ in your console for more details.

AGENDA

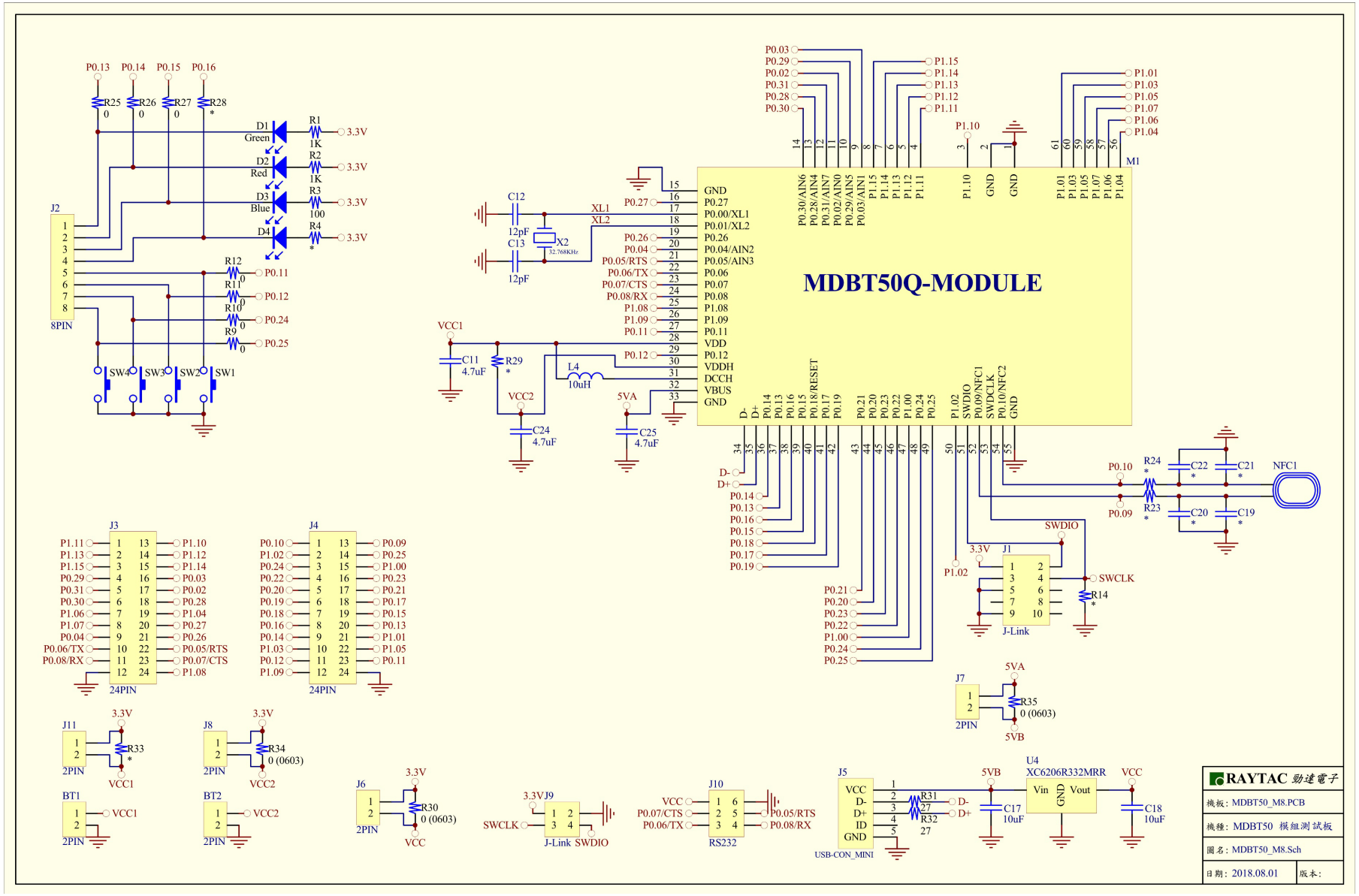
- MDBT50Q-DB-40 Walk Around
- nRF52840/MDBT50Q GPIO



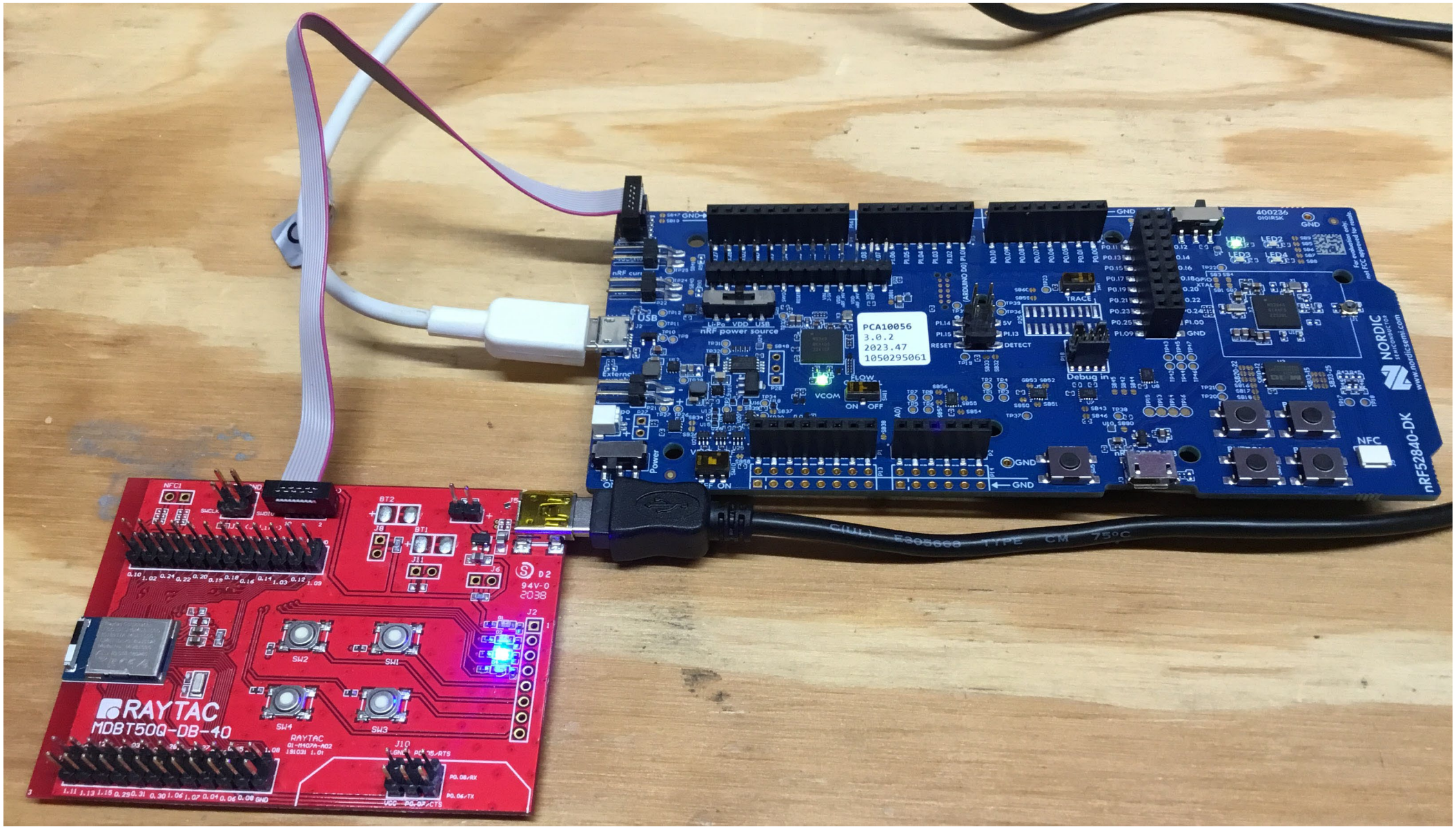
MDBT50Q-DB-40



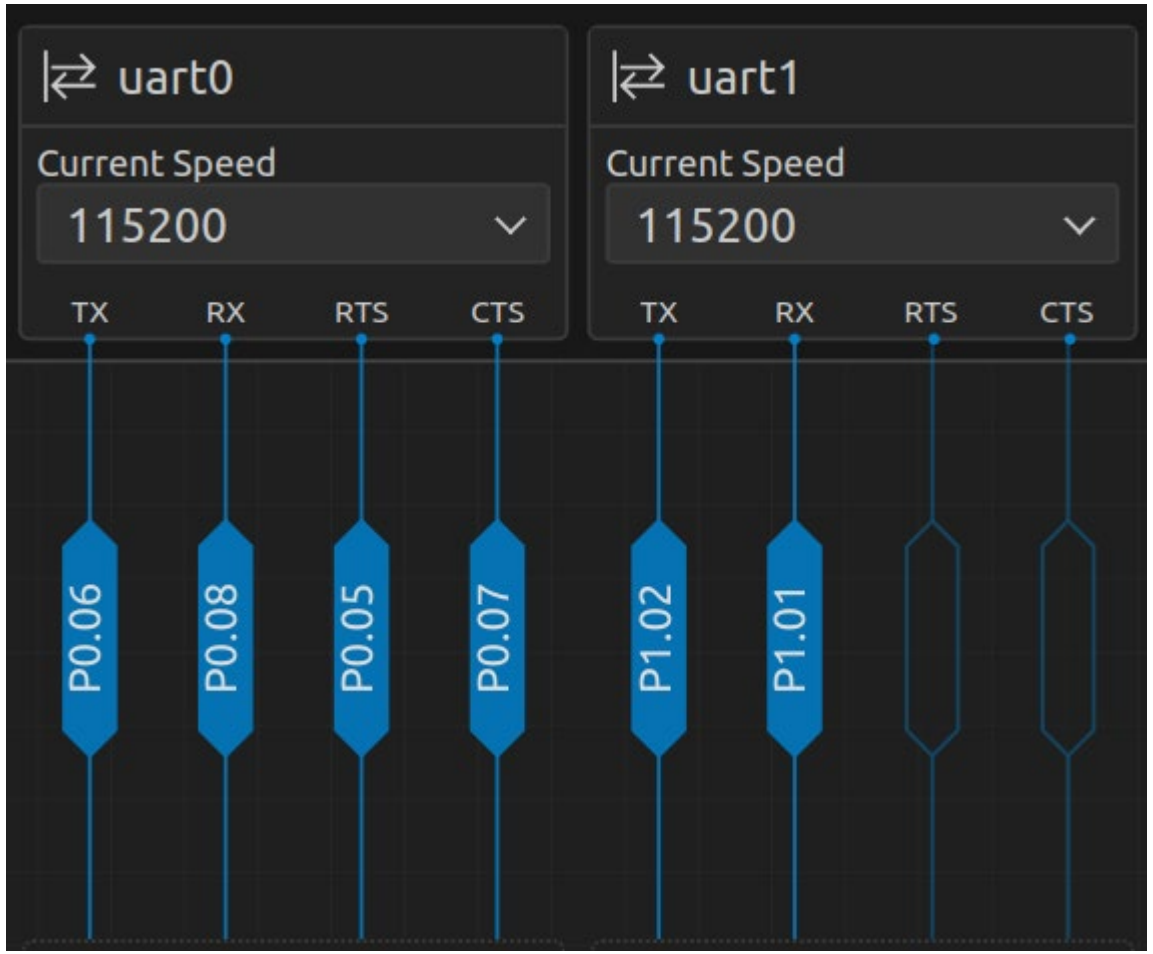
MDBT50Q-DB-40



MDBT50Q-DB-40



Devicetree – UART0 and UART1

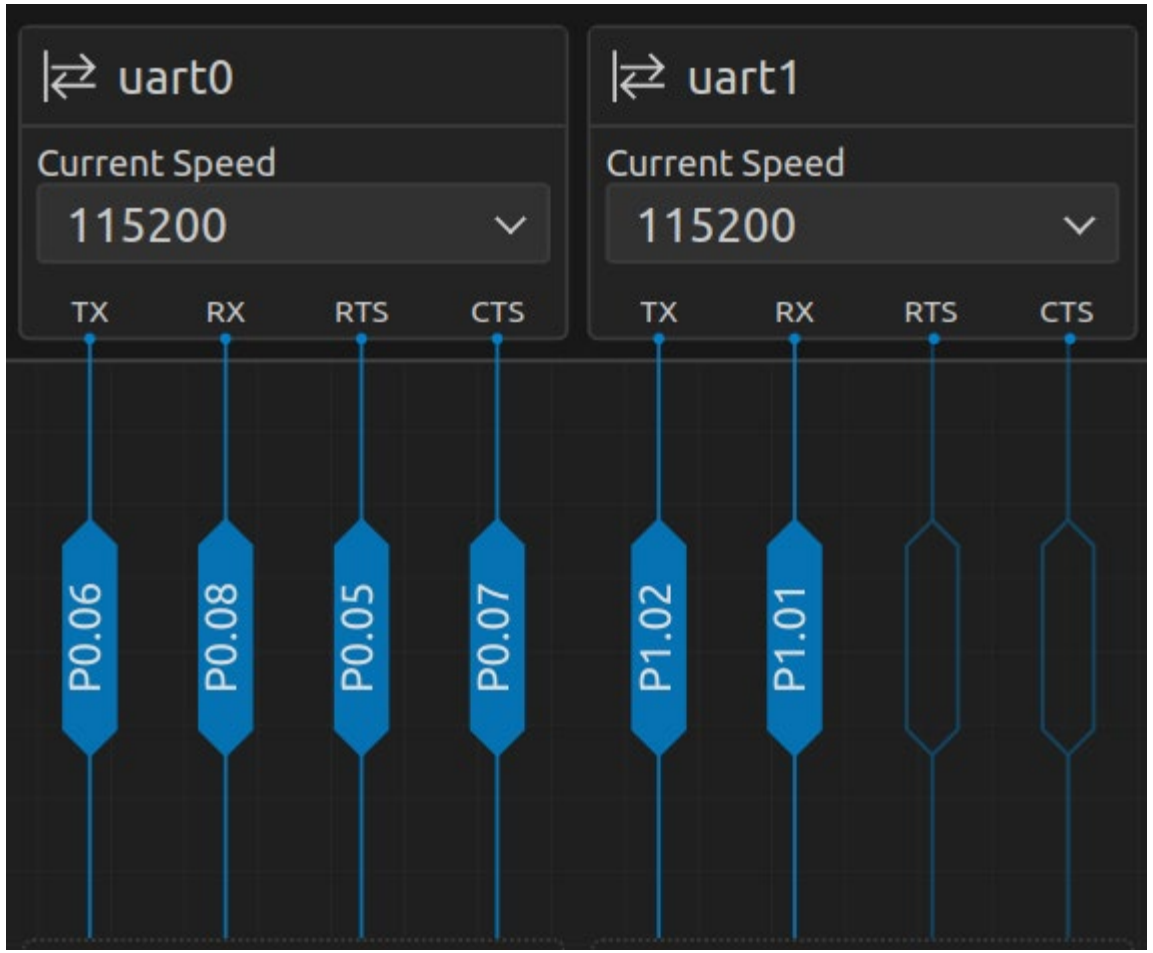



```
&uart0 {
compatible = "nordic,nrf-uarte";
status = "okay";
current-speed = <115200>;
pinctrl-0 = <&uart0_default>;
pinctrl-1 = <&uart0_sleep>;
pinctrl-names = "default", "sleep";
};
```

```
&uart1 {
compatible = "nordic,nrf-uarte";
status = "okay";
current-speed = <115200>;
pinctrl-0 = <&uart1_default>;
pinctrl-1 = <&uart1_sleep>;
pinctrl-names = "default", "sleep";
};
```

Needs to be 9600

Pinctrl – UART0 and UART1




```
&pinctrl {
    uart0_default: uart0_default {
        group1 {
            psels = <NRF_PSEL(UART_TX, 0, 6)>,
                <NRF_PSEL(UART_RTS, 0, 5)>;
        };
        group2 {
            psels = <NRF_PSEL(UART_RX, 0, 8)>,
                <NRF_PSEL(UART_CTS, 0, 7)>;
            bias-pull-up;
        };
    };

    uart0_sleep: uart0_sleep {
        group1 {
            psels = <NRF_PSEL(UART_TX, 0, 6)>,
                <NRF_PSEL(UART_RX, 0, 8)>,
                <NRF_PSEL(UART_RTS, 0, 5)>,
                <NRF_PSEL(UART_CTS, 0, 7)>;
            low-power-enable;
        };
    };

    uart1_default: uart1_default {
        group1 {
            psels = <NRF_PSEL(UART_RX, 1, 1)>;
            bias-pull-up;
        };
        group2 {
            psels = <NRF_PSEL(UART_TX, 1, 2)>;
        };
    };

    uart1_sleep: uart1_sleep {
        group1 {
            psels = <NRF_PSEL(UART_RX, 1, 1)>,
                <NRF_PSEL(UART_TX, 1, 2)>;
            low-power-enable;
        };
    };
```

UART0 LED Control / MaxBotix Range Finder

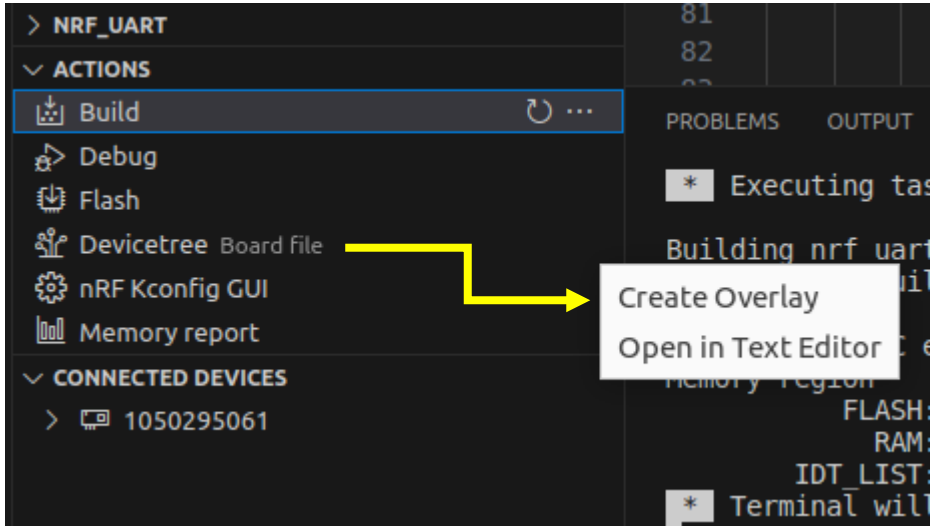
```
// Define receive buffer sizes and receive timeout
#define RECEIVE_BUF0_SIZE 16              // UART0 Receive Buffer Size - LED Control
#define RECEIVE_BUF1_SIZE 16              // UART1 Receive Buffer Size - MaxBotix Range Finder
#define UART1_RX_MSG_QUEUE_SIZE 16        // Maximum number of UART1 Receive Buffers
#define RECEIVE_TIMEOUT 100               // Receive timeout in mS

struct uart1_msg_queue_prototype
{
  uint8_t bites[RECEIVE_BUF1_SIZE];
  uint32_t len;
};

struct uart1_msg_queue_prototype new_msg;                 // Used in MaxBotix UART1 Callback
struct uart1_msg_queue_prototype incoming_msg;            // MaxBotix Raw Incoming UART1 Data

uint8_t uart1_double_buffer[2][RECEIVE_BUF1_SIZE];        // MaxBotix Receive Buffers
uint8_t *uart1_buf_next = uart1_double_buffer[1];
uint8_t incoming_buf[RECEIVE_BUF1_SIZE + 1];              // MaxBotix Data Buffer
uint8_t bufindx;                                          // MaxBotix Buffer Array Index
uint8_t indx;                                             // MaxBotix Buffer Array Index

K_MSGQ_DEFINE(uart1_rx_msgq,sizeof(struct uart1_msg_queue_prototype),UART1_RX_MSG_QUEUE_SIZE, 1);
```

UART0 LED Control / MaxBotix Range Finder

```
// Get the device pointers of the LEDs through gpio_dt_spec
static const struct gpio_dt_spec led0 = GPIO_DT_SPEC_GET(DT_ALIAS(led0), gpios);
static const struct gpio_dt_spec led1 = GPIO_DT_SPEC_GET(DT_ALIAS(led1), gpios);
static const struct gpio_dt_spec led2 = GPIO_DT_SPEC_GET(DT_ALIAS(led2), gpios);

// Get the device pointer of the UART hardware
const struct device *uart_0 = DEVICE_DT_GET(DT_NODELABEL(uart0));
const struct device *uart_1 = DEVICE_DT_GET(DT_NODELABEL(uart1));

// Define the transmission buffer, which is a buffer to hold the data to be sent over UART0
static uint8_t tx_buf0[] = {"\r\nBLE Application Development Using Raytac Modules\r\n"
"\r\nPress 1-3 on your keyboard to toggle LEDS 1-3 on the MDBT50Q-DB-40 dev board\r\n"
"Press ENTER to extinguish all LEDS\r\n"};

// Define the UART0 LED Control receive buffer
static uint8_t rx_buf0[RECEIVE_BUF0_SIZE] = {0};
```

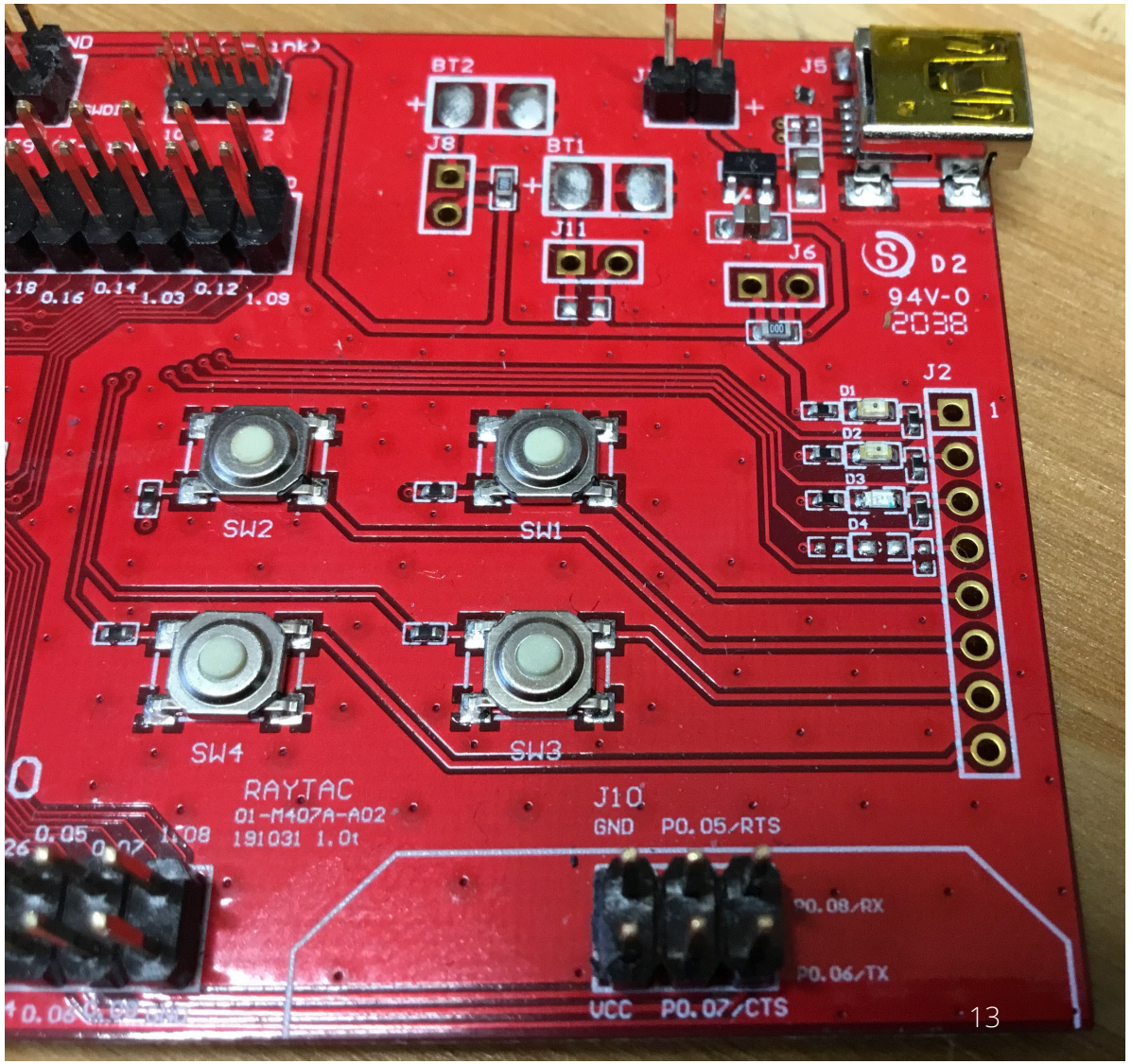
MaxBotix Range Finder – Operating on UART1

```
// raytac_mdbt50q_db_40_nrf52840.overlay
// This overlay will override the default 115200 baud rate for UART1

// To get started, press Ctrl+Space to bring up the completion menu and view the available nodes.

// You can also use the buttons in the sidebar to perform actions on nodes.
// Actions currently available include:

// * Enabling / disabling the node
// * Adding the bus to a bus
// * Removing the node
// * Connecting ADC channels

// For more help, browse the DeviceTree documentation at https://docs.zephyrproject.org/latest/guides/dts/index.html
// You can also visit the nRF DeviceTree extension documentation at https://nrfconnect.github.io/vscode-nrf-
// connect/devicetree/nrfdevicetree.html

&uart1 {
compatible = "nordic,nrf-uarte";
status = "okay";
current-speed = <9600>;
pinctrl-0 = <&uart1_default>;
pinctrl-1 = <&uart1_sleep>;
pinctrl-names = "default", "sleep";
};
```



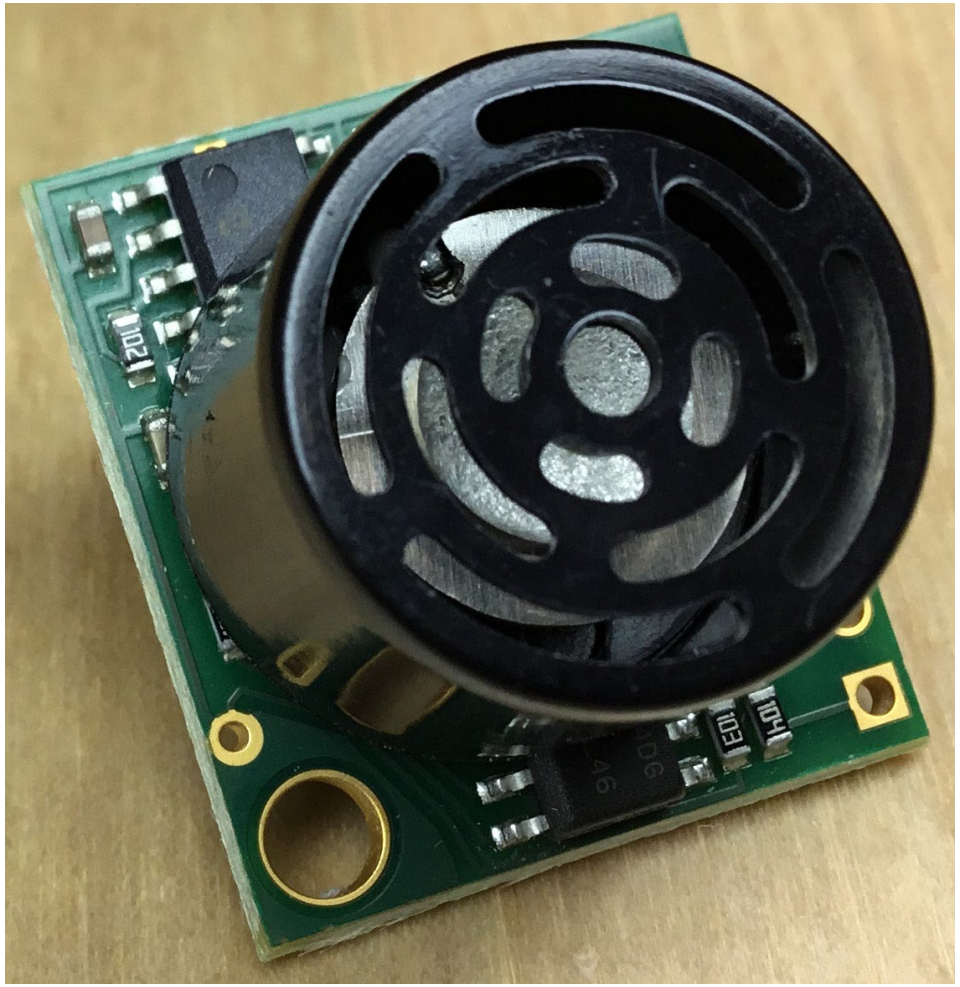
UART0/LED Control Callback Function

```
// Define the callback function for UART0 - LED Control
static void uart0_cb(const struct device *dev, struct uart_event *evt0, void *user_data)
{
  switch (evt0->type) {

  case UART_RX_RDY:
   switch(evt0->data.rx.len)
   {
     case 1:
      switch(evt0->data.rx.buf[evt0->data.rx.offset])
      {
        case '1':
         gpio_pin_toggle_dt(&led0);
        break;
        case '2':
         gpio_pin_toggle_dt(&led1);
        break;
        case '3':
         gpio_pin_toggle_dt(&led2);
        break;
        case '\r':
         gpio_pin_set_dt(&led0,0);
         gpio_pin_set_dt(&led1,0);
         gpio_pin_set_dt(&led2,0);
        break;
      }
  break;
}
  case UART_RX_DISABLED:
   uart_rx_enable(dev ,rx_buf0,sizeof rx_buf0,RECEIVE_TIMEOUT);
  break;
  default:
  break;
 }
}
```



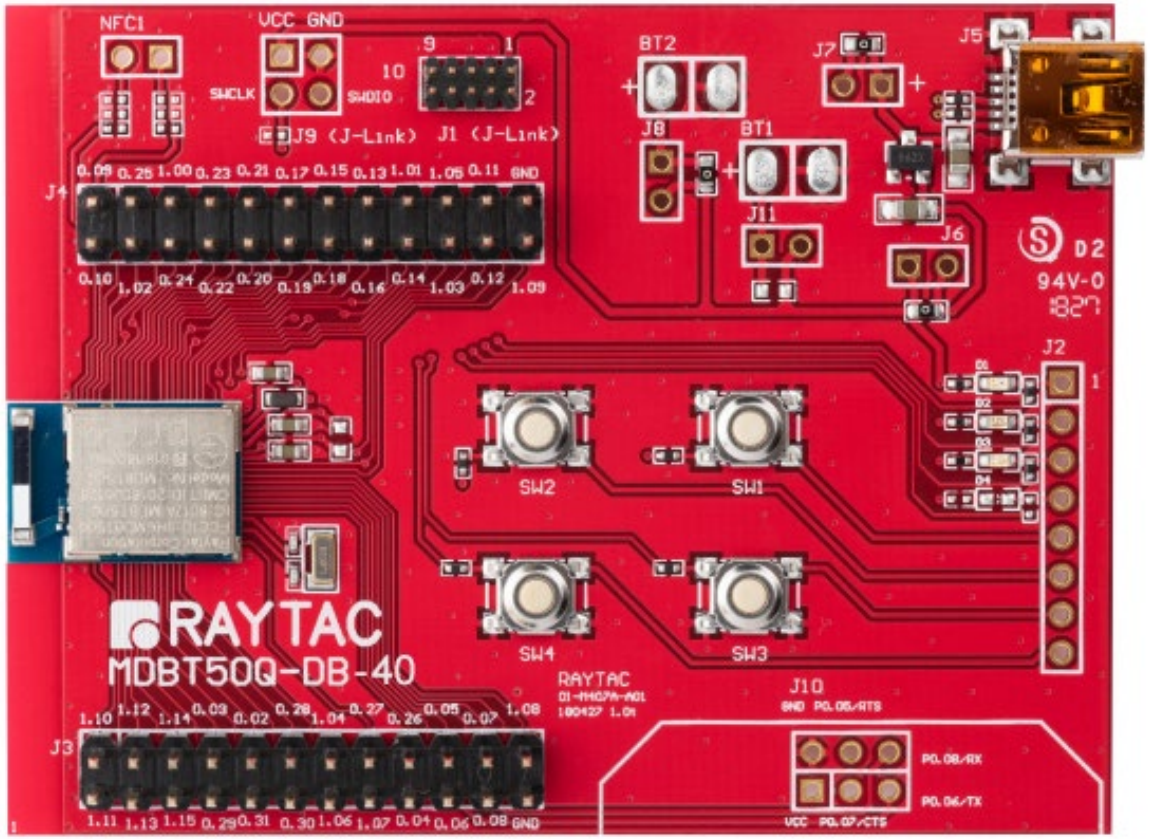
MaxBotix Range Finder Callback Function

```
// Define the callback function for UART1 – MaxBotix Range Finder
static void uart1_cb(const struct device *dev, struct uart_event *evt1, void *user_data)
{
  switch (evt1->type) {

  case UART_RX_RDY:
   memcpy(new_msg.bites, evt1->data.rx.buf + evt1->data.rx.offset, evt1->data.rx.len);
   new_msg.len = evt1->data.rx.len;
   if(k_msgq_put(&uart1_rx_msgq, &new_msg, K_NO_WAIT) != 0)
   {
    printk("Error: UART RX msg queue full\r\n");
   }
  break;
  case UART_RX_BUF_REQUEST:
   uart_rx_buf_rsp(uart_1,uart1_buf_next,RECEIVE_BUF1_SIZE);
  break;
  case UART_RX_BUF_RELEASED:
   uart1_buf_next = evt1->data.rx_buf.buf;
  break;
  case UART_RX_DISABLED:
   uart_rx_enable(uart_1 ,uart1_double_buffer[0], RECEIVE_BUF1_SIZE, RECEIVE_TIMEOUT);
  break;
  default:
  break;
 }
}
```



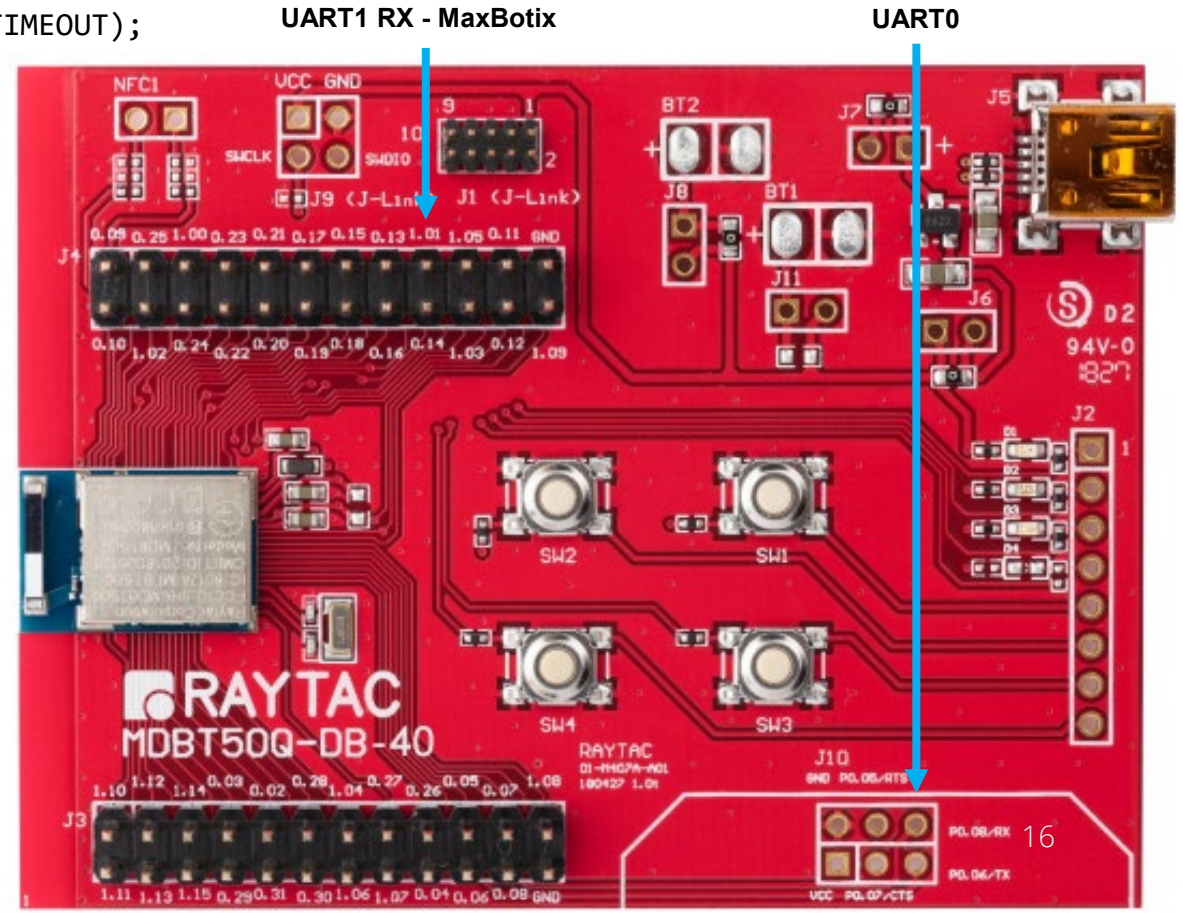
Configure UART0 LED Control GPIO

```
// Verify that the LED Control UART0 device is ready
if (!device_is_ready(uart_0)){
  printk("UART0 device not ready\r\n");
  return 1 ;
}
// Verify that the MaxBotix UART1 device is ready
if (!device_is_ready(uart_1)){ // MaxBotix
  printk("UART1 device not ready\r\n");
  return 1 ;
}
// Verify that the LED devices are ready
if (!device_is_ready(led0.port)){
  printk("GPIO device is not ready\r\n");
  return 1;
}
// Configure the GPIOs of the LEDs
gpio_pin_configure_dt(&led0, GPIO_OUTPUT_ACTIVE);
gpio_pin_configure_dt(&led1, GPIO_OUTPUT_ACTIVE);
gpio_pin_configure_dt(&led2, GPIO_OUTPUT_ACTIVE);
```

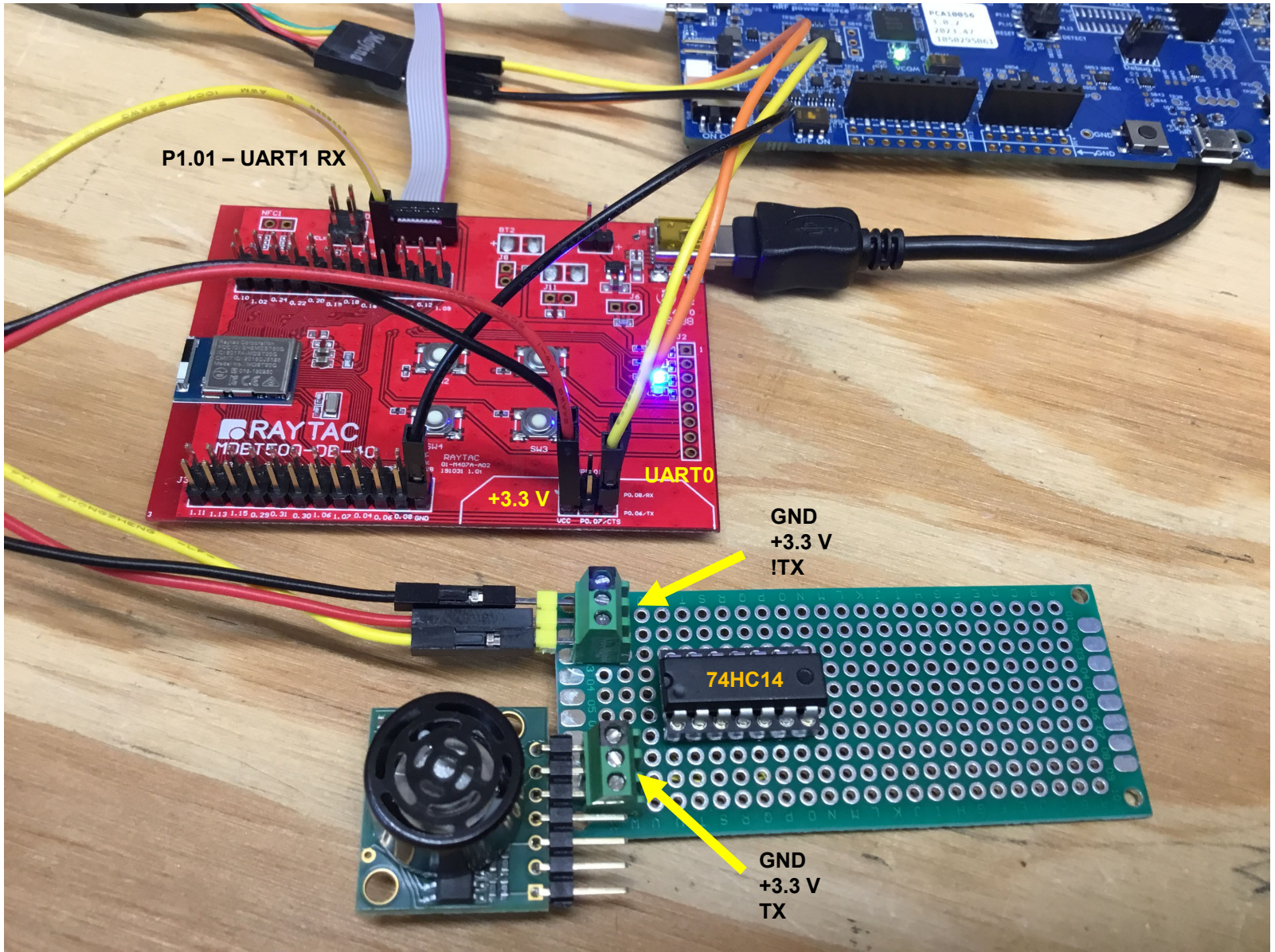


Enable UART0 and UART1

```
// Register the UART callback functions
uart_callback_set(uart_0, uart0_cb, NULL); // LED Control
uart_callback_set(uart_1, uart1_cb, NULL); // MaxBotix Range Finder
// Send the banner message
uart_tx(uart_0, tx_buf0, sizeof(tx_buf0), SYS_FOREVER_MS);
// Start receiving by calling uart_rx_enable() and pass it the address of the receive buffer
uart_rx_enable(uart_0 ,rx_buf0,sizeof rx_buf0,RECEIVE_TIMEOUT);
uart_rx_enable(uart_1 ,uart1_double_buffer[0], RECEIVE_BUF1_SIZE, RECEIVE_TIMEOUT);
// Retrieve and process the incoming UART1 data
    while (1) {
        k_msgq_get(&uart1_rx_msgq, &incoming_msg, K_FOREVER);

        for(indx = 0; indx < incoming_msg.len; indx++)
        {
            incoming_buf[bufindx++] = incoming_msg.bites[indx];
        }

        for(indx = 0; indx < bufindx; indx++)
        {
            if(incoming_buf[indx] == 0x0D)
            {
                bufindx = 0;
                printk("%s\r\n",incoming_buf);
            }
        }
    }
}
```

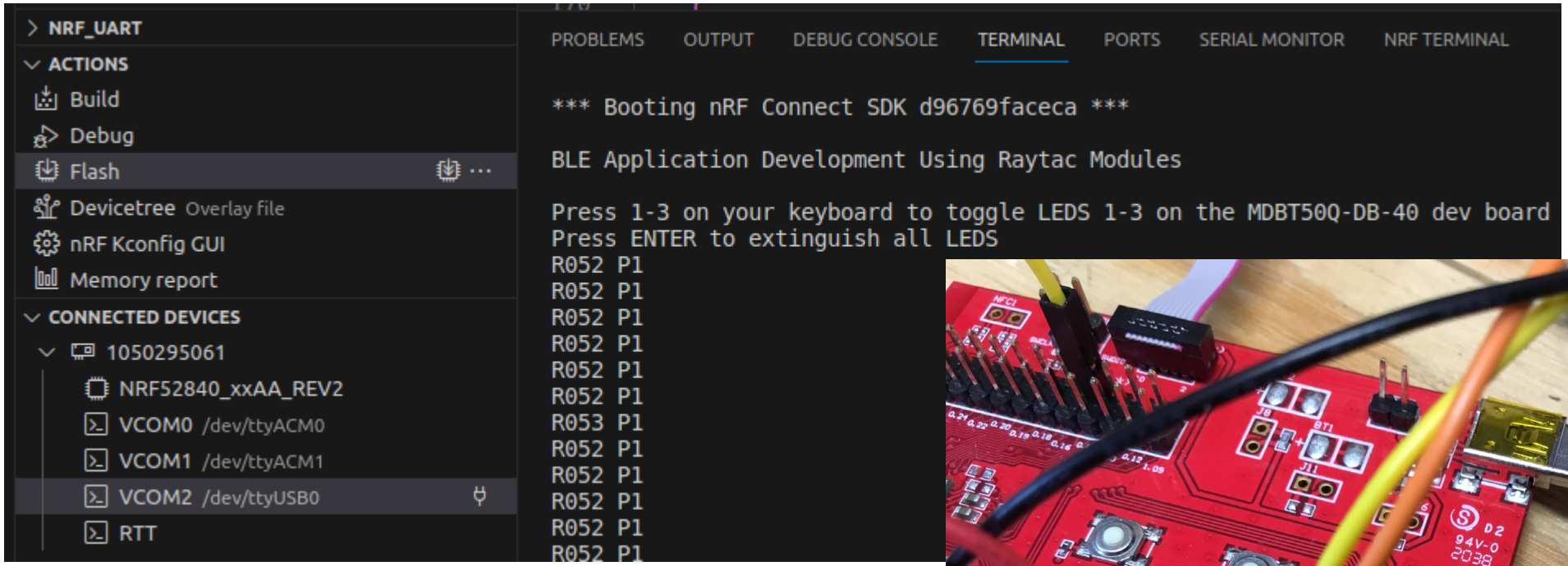


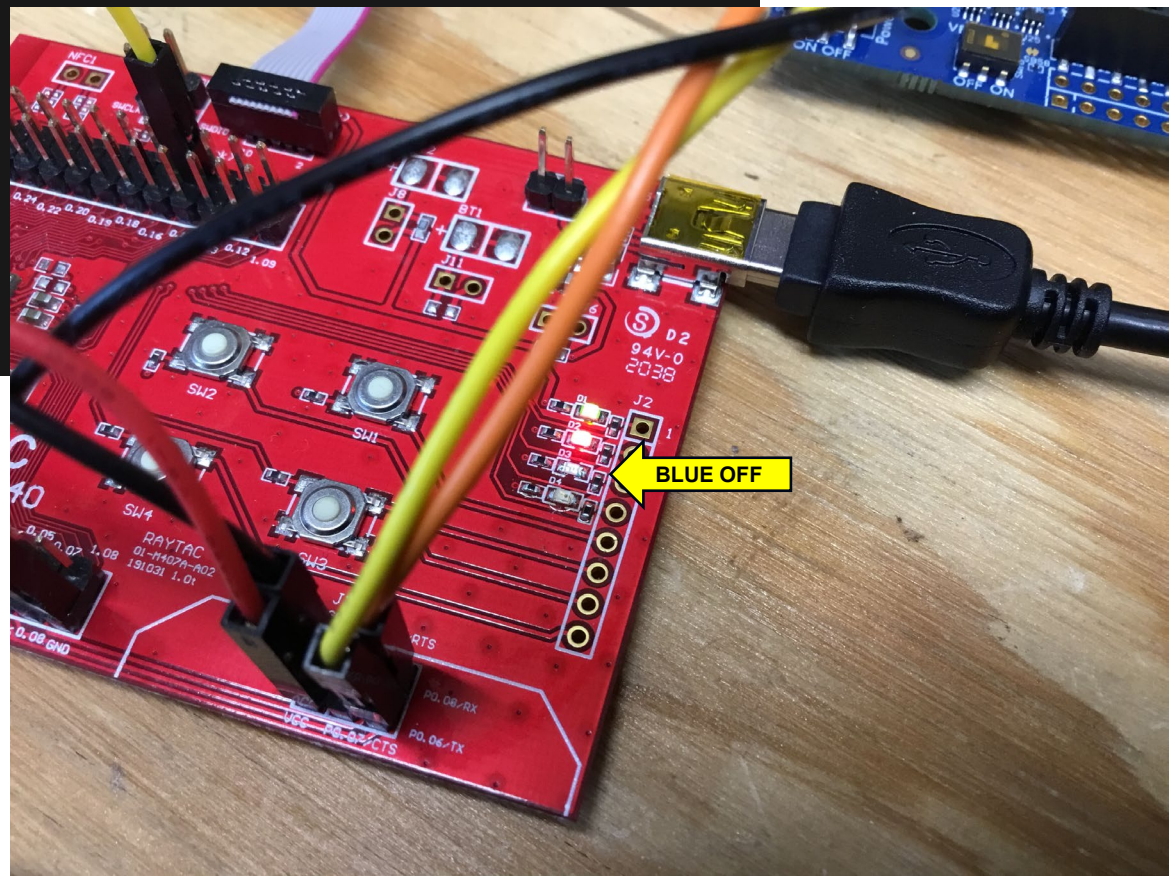
Fire It Up



MaxBotix Range Packet

Fire It Up


NRF_UART
ACTIONS
Build
Debug
Flash
Devicetree Overlay file
nRF Kconfig GUI
Memory report
CONNECTED DEVICES
1050295061
NRF52840_xxAA_REV2
VCOM0 /dev/ttyACM0
VCOM1 /dev/ttyACM1
VCOM2 /dev/ttyUSB0
RTT
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SERIAL MONITOR
NRF TERMINAL
*** Booting nRF Connect SDK d96769faceca ***
BLE Application Development Using Raytac Modules
Press 1-3 on your keyboard to toggle LEDS 1-3 on the MDBT50Q-DB-40 dev board
Press ENTER to extinguish all LEDS
R052 P1
R052 P1
R052 P1
R052 P1
R052 P1
R052 P1
R053 P1
R052 P1
R052 P1
R052 P1
R052 P1
R052 P1


BLUE OFF

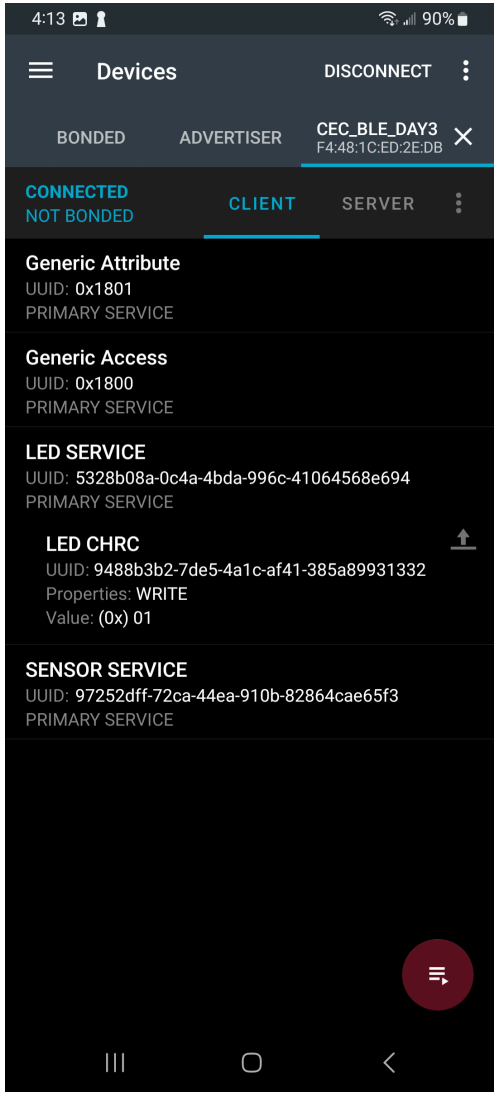
Next Time…

Thank you for attending!!!

Please consider the resources below:

- Today's Download Package
- nordicsemi.com
- nRF52840 User Guide
- raytac.com
- maxbotix.com

MORE TO COME..



CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets