



Constructing a Raspberry Pi RP2040 Low-Power Sensor Node

Day 4:

Data Radio Rodeo

Sponsored by



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.

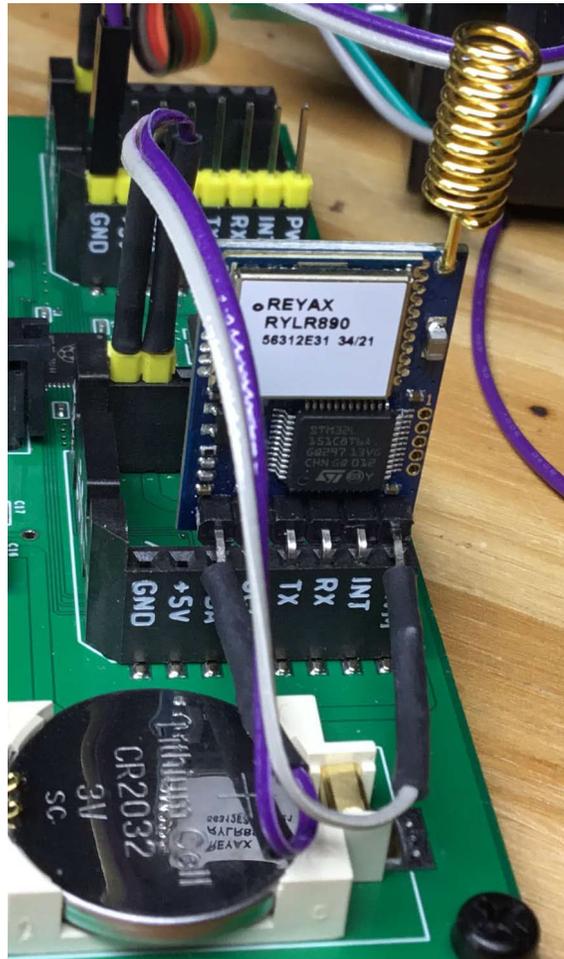


Fred Eady

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

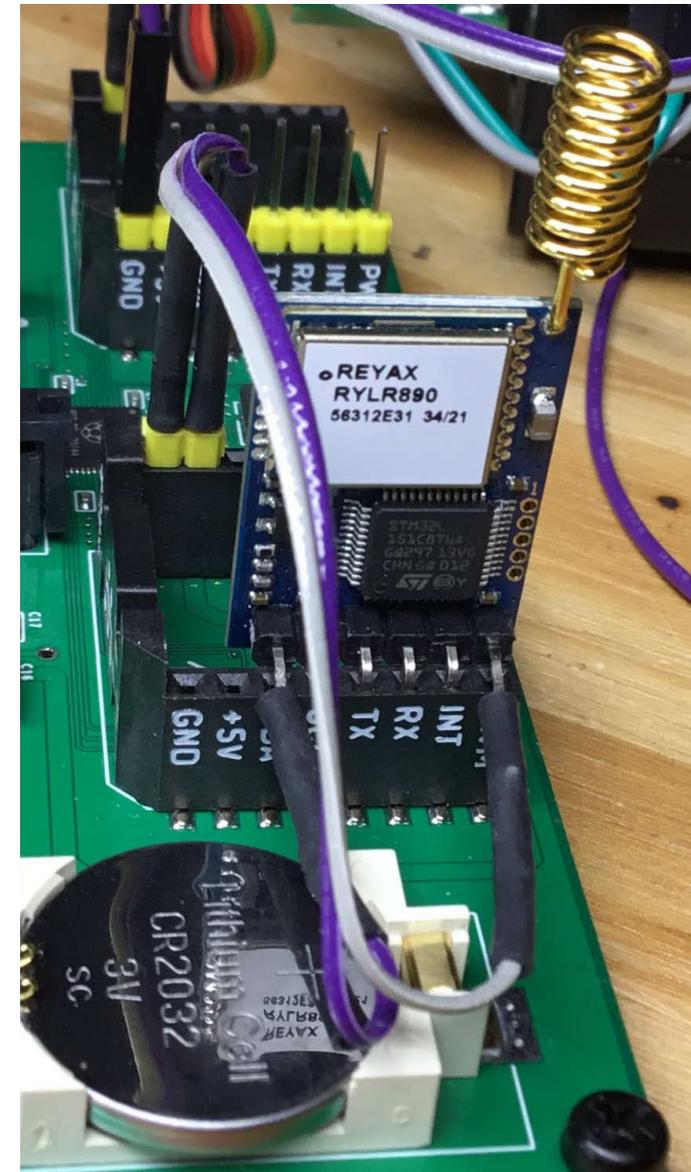
AGENDA

- **LORA**
- **XBee**



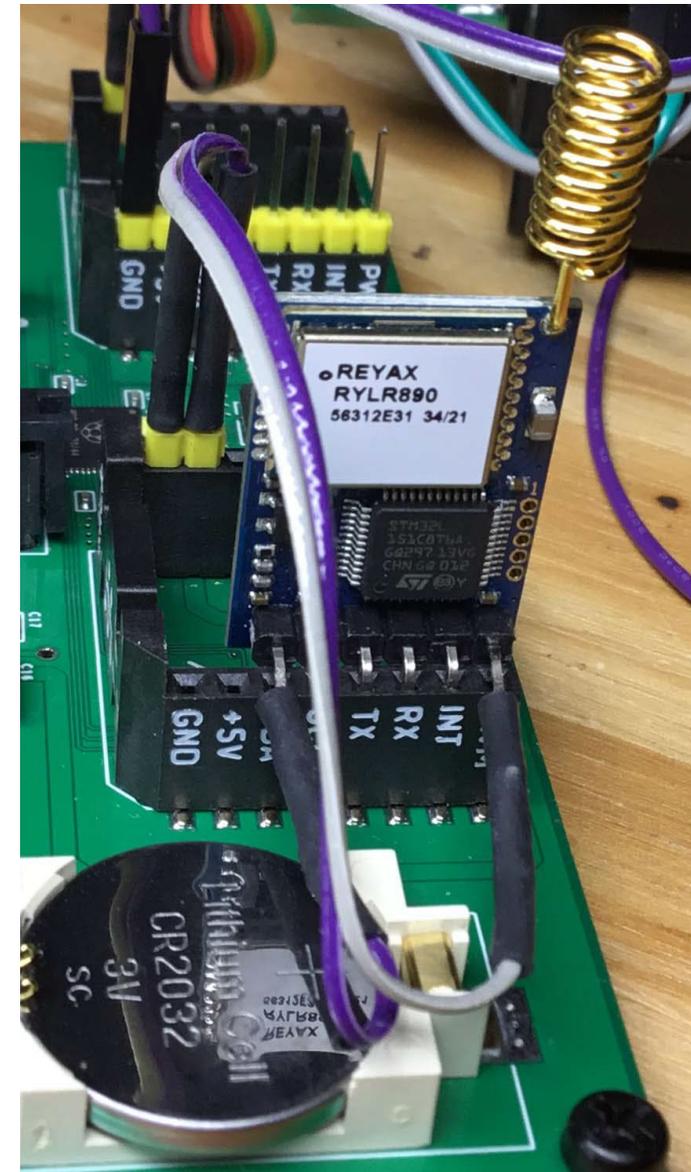
REYAX RYLR896 – CMakeLists.txt

```
M CMakeLists.txt
1 # Set minimum required version of CMake
2 cmake_minimum_required(VERSION 3.12)
3
4 # Include build functions from Pico SDK
5 include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)
6
7 # Set name of project (as PROJECT_NAME) and C/C++ standards
8 project(reyax C CXX ASM)
9 set(CMAKE_C_STANDARD 11)
10 set(CMAKE_CXX_STANDARD 17)
11
12 # Creates a pico-sdk subdirectory in our project for the libraries
13 pico_sdk_init()
14
15 # Tell CMake where to find the executable source file
16 add_executable(${PROJECT_NAME}
17     uart_advanced.c
18 )
19
20 # Create map/bin/hex/uf2 files
21 pico_add_extra_outputs(${PROJECT_NAME})
22
23 # Link to pico_stdlib (gpio, time, etc. functions)
24 target_link_libraries(${PROJECT_NAME}
25     pico_stdlib
26     hardware_gpio
27     hardware_uart
28     hardware_irq
29 )
30
31 # Enable usb output, enable uart output
32 pico_enable_stdio_usb(${PROJECT_NAME} 0)
33 pico_enable_stdio_uart(${PROJECT_NAME} 1)
```



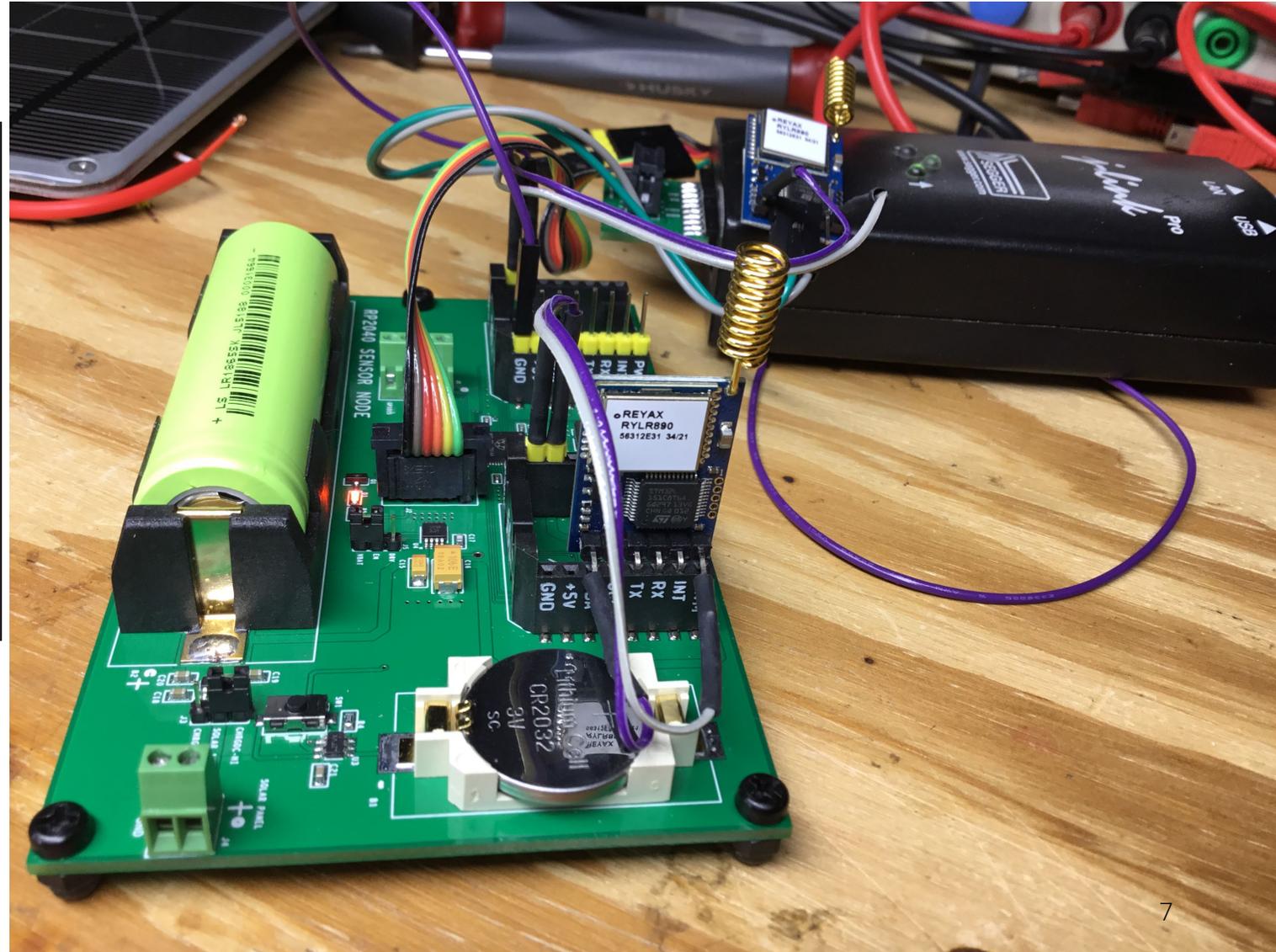
REYAX RYLR896 – Interrupt Handler, Variables and GPIO Pin Assignments

```
5 // Application States
6 enum
7 {
8     wait4ready,
9     sendAT,
10    sendMsg
11 };
12 // Serial Input/Output Monitor Pins
13 // REYAX Receiver Pins
14 #define UART0_TX_PIN    0
15 #define UART0_RX_PIN    1
16 // REYAX Transmitter Pins
17 #define UART1_TX_PIN    20
18 #define UART1_RX_PIN    21
19 // REYAX Reset Pin
20 #define NRST1_PIN       25
21 // Sensor Node LED Pin
22 #define LED_PIN         17
23 // DONE Pin
24 #define DONE_PIN       16
25
26 uint8_t rxBuf_1[16];    // uart1 RX buffer
27 uint8_t indx_1;        // RX buffer index
28 uint8_t state;         // state machine variable
29
30 // RX interrupt handler
31 void uart1_rx_handler()
32 {
33     while (uart_is_readable(uart1))
34     {
35         rxBuf_1[indx_1++] = uart_getc(uart1);
36     }
37 }
```



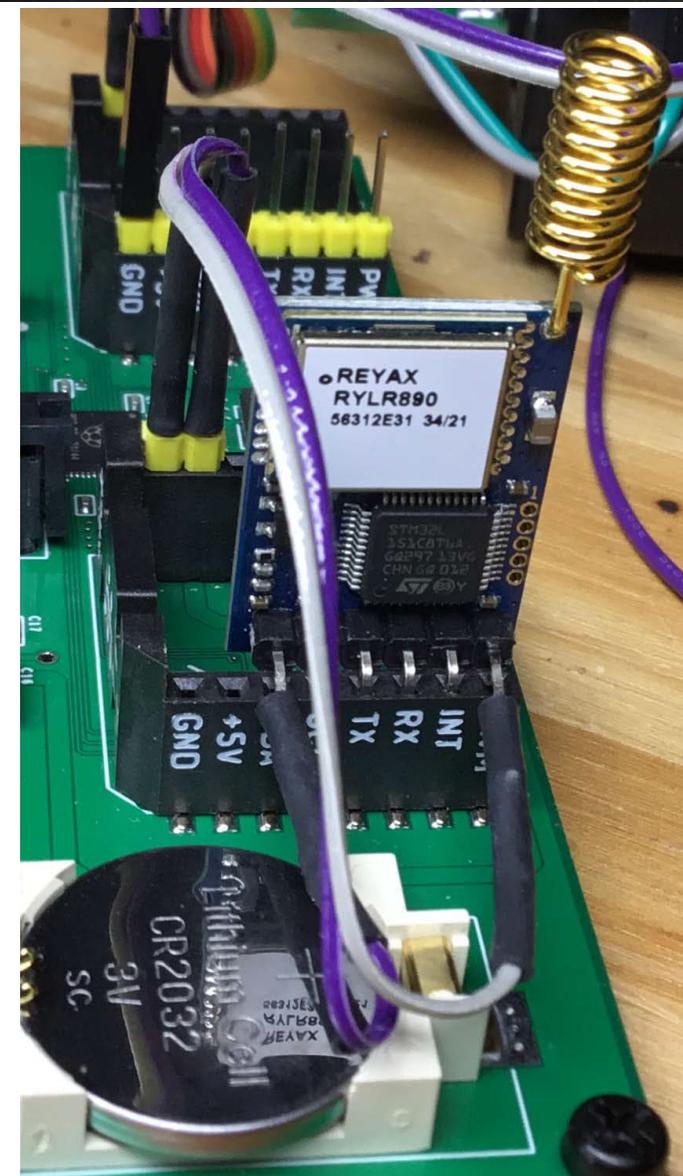
REYAX RYLR896 – GPIO Initialization

```
39 int main()
40 {
41     uint8_t i;
42
43     // GPIO Init
44     gpio_init(NRST1_PIN);
45     gpio_set_dir(NRST1_PIN, GPIO_OUT);
46     gpio_put(NRST1_PIN, false);
47
48     gpio_init(LED_PIN);
49     gpio_set_dir(LED_PIN, GPIO_OUT);
50     gpio_put(LED_PIN, false);
51
52     gpio_init(DONE_PIN);
53     gpio_set_dir(DONE_PIN, GPIO_OUT);
54     gpio_put(DONE_PIN, false);
```



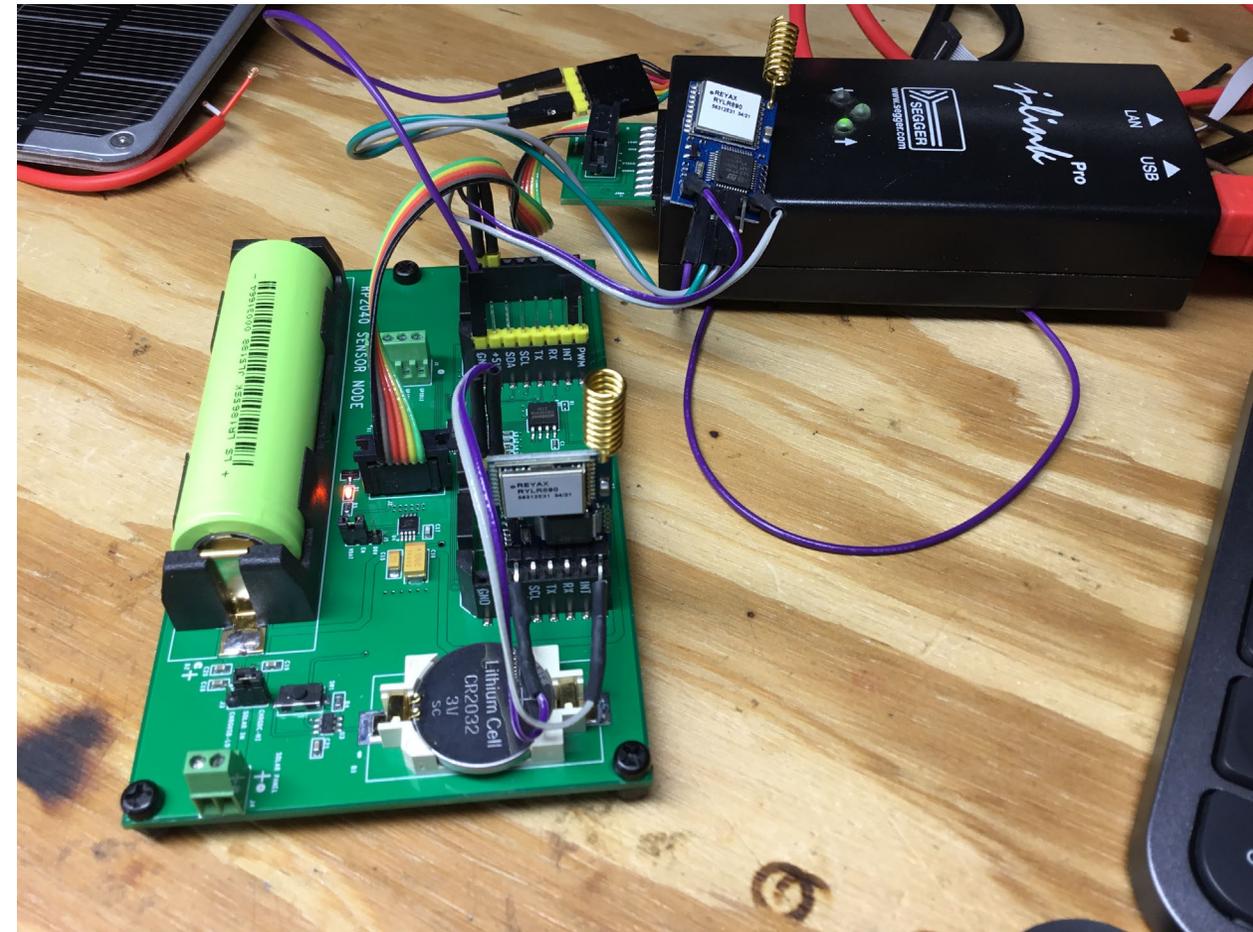
REYAX RYLR896 – UART Initialization

```
56 // UART Init
57 stdio_uart_init_full(uart0, 115200, UART0_TX_PIN, UART0_RX_PIN);
58 stdio_uart_init_full(uart1, 115200, UART1_TX_PIN, UART1_RX_PIN);
59
60 // Turn off FIFO
61 uart_set_fifo_enabled(uart1, false);
62
63 // Set up and enable the interrupt handler
64 irq_set_exclusive_handler(UART1_IRQ, uart1_rx_handler);
65 irq_set_enabled(UART1_IRQ, true);
66
67 // Enable the UART RX interrupt
68 uart_set_irq_enables(uart1, true, false);
```



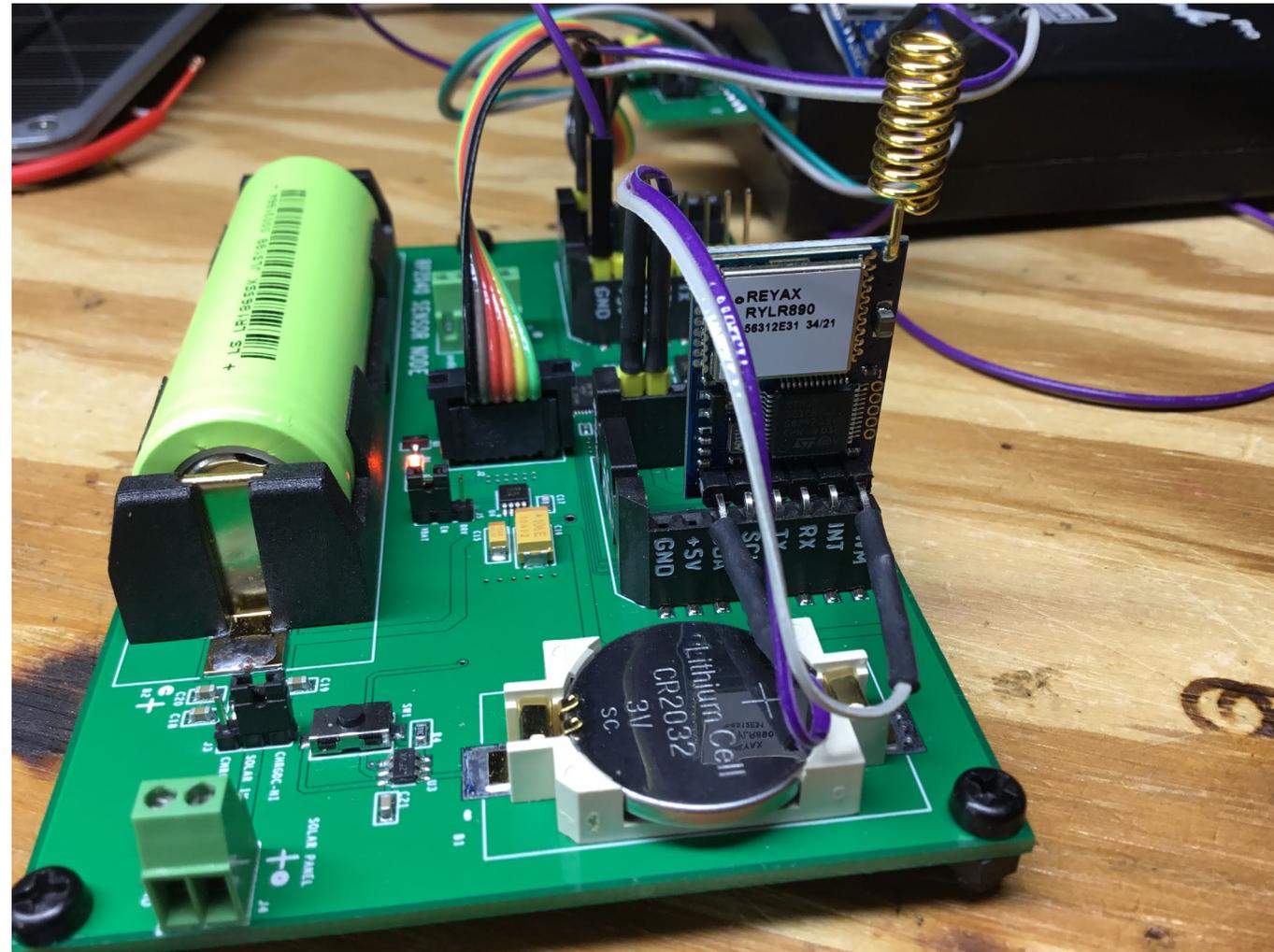
REYAX RYLR896 – Lift Reset Pin and Begin

```
70 // Init state machine variable and init RX buffer index
71 state = wait4ready;
72 indx_1 = 0;
73 // Pull REYAX out of reset
74 sleep_ms(100);
75 gpio_put(NRST1_PIN, true);
```



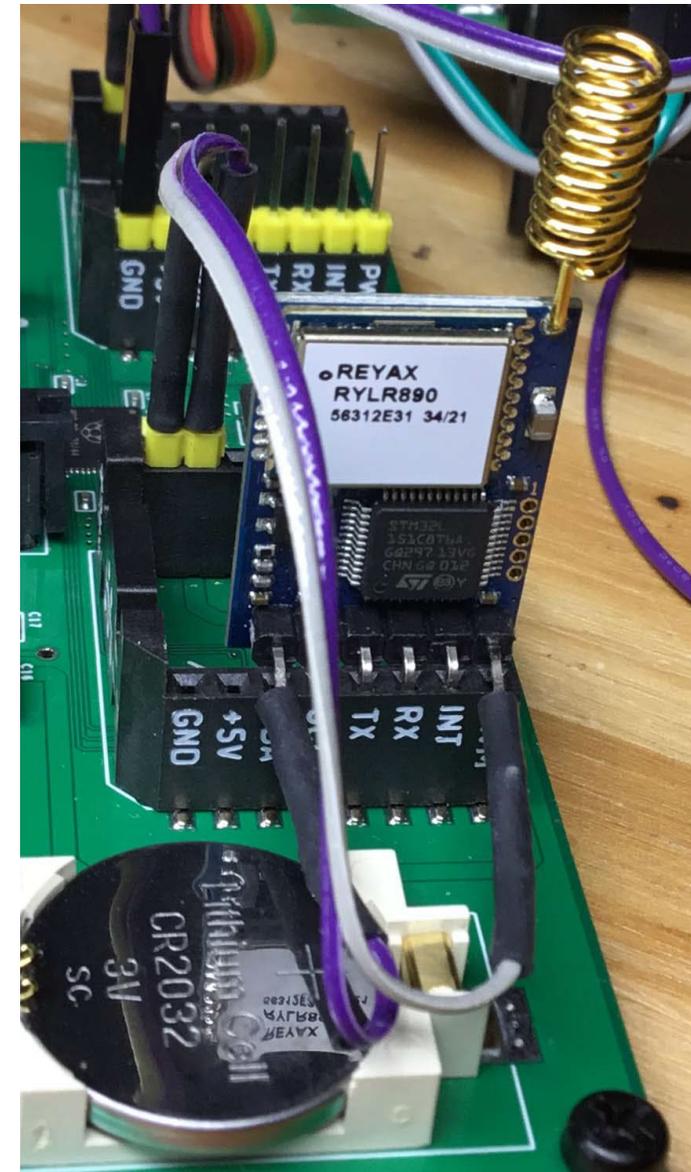
REYAX RYLR896 – wait4ready

```
77 while (1)
78 {
79     switch(state)
80     {
81         // Look for +READY from the REYAX and discard it
82         case wait4ready:
83             if(indx_1 > 0)
84             {
85                 sleep_ms(100);
86                 // DEBUG MONITOR
87                 for(i=0;i<indx_1;i++)
88                 {
89                     uart_putc(uart0, rxBuf_1[i]);
90                 }
91                 // DEBUG MONITOR END
92                 indx_1 = 0;
93                 state = sendAT;
94             }
95             break;
```



REYAX RYLR896 - sendAT

```
96 // Send AT command
97 case sendAT:
98     uart_puts(uart1, "AT\r\n");
99     while(indx_1 == 0);
100    sleep_ms(100);
101    // DEBUG MONITOR START
102    for(i=0;i<indx_1;i++)
103    {
104        uart_putc(uart0, rxBuf_1[i]);
105    }
106    // DEBUG MONITOR END
107    if(rxBuf_1[indx_1-3] == 'K')
108    {
109        // DEBUG MONITOR START
110        uart_putc(uart0, rxBuf_1[indx_1 - 3]);
111        // DEBUT MONITOR END
112        indx_1 = 0;
113        state = sendMsg;
114    }
115 break;
```

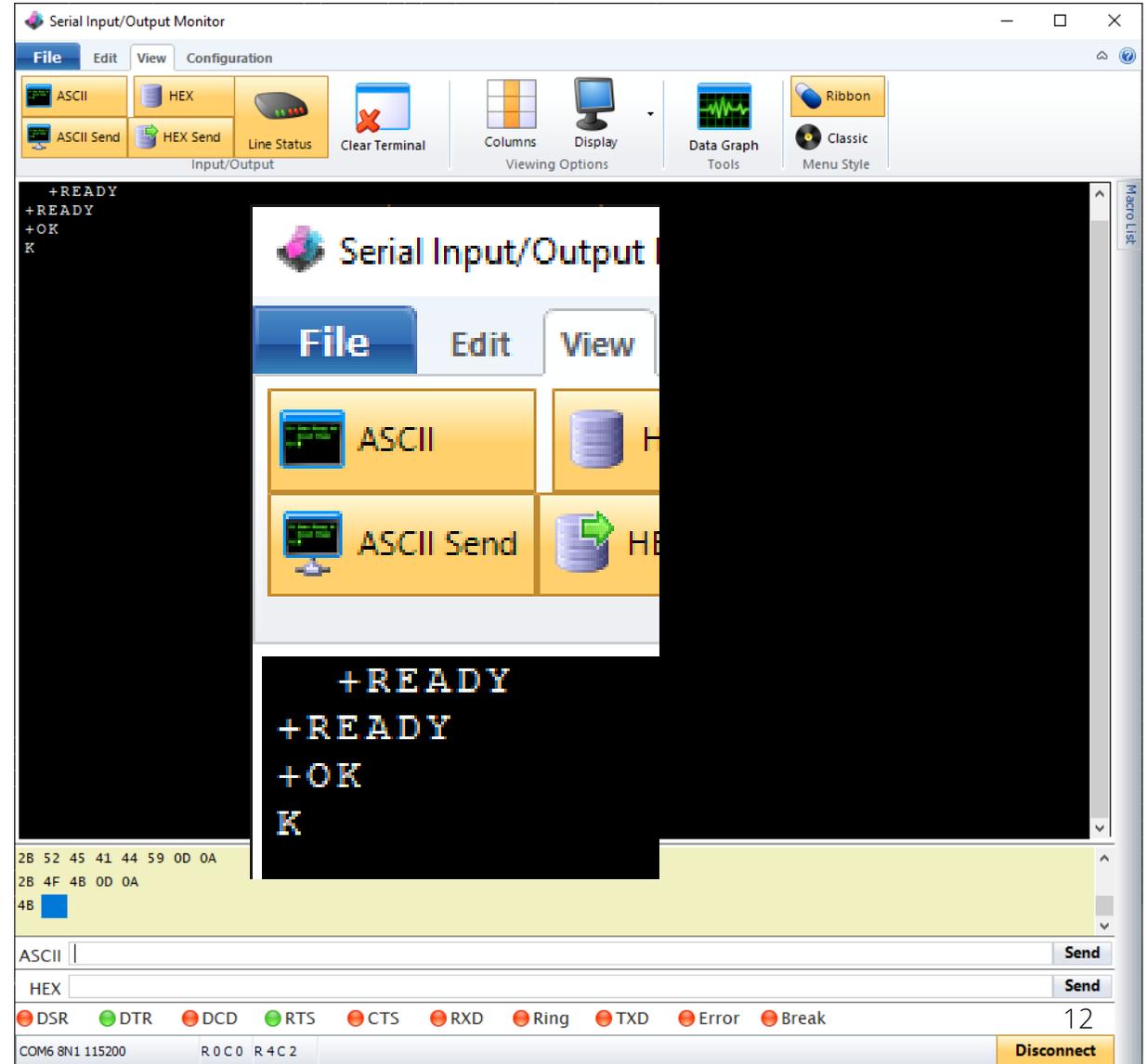


REYAX RYLR896 - sendAT

```

96 // Send AT command
97 case sendAT:
98     uart_puts(uart1, "AT\r\n");
99     while(indx_1 == 0);
100     sleep_ms(100);
101     // DEBUG MONITOR START
102     for(i=0;i<indx_1;i++)
103     {
104         uart_putc(uart0, rxBuf_1[i]);
105     }
106     // DEBUG MONITOR END
107     if(rxBuf_1[indx_1-3] == 'K')
108     {
109         // DEBUG MONITOR START
110         uart_putc(uart0, rxBuf_1[indx_1 - 3]);
111         // DEBUT MONITOR END
112         indx_1 = 0;
113         state = sendMsg;
114     }
115     break;

```



The screenshot displays the Serial Input/Output Monitor application interface. The main window shows a terminal window with the following output:

```

+READY
+READY
+OK
K

```

The application has a ribbon interface with the following tabs: File, Edit, View, Configuration. The ribbon includes the following groups and items:

- Input/Output:** ASCII, HEX, ASCII Send, HEX Send, Line Status, Clear Terminal.
- Viewing Options:** Columns, Display.
- Tools:** Data Graph.
- Menu Style:** Ribbon, Classic.

The terminal window also shows a hex dump of the received data:

```

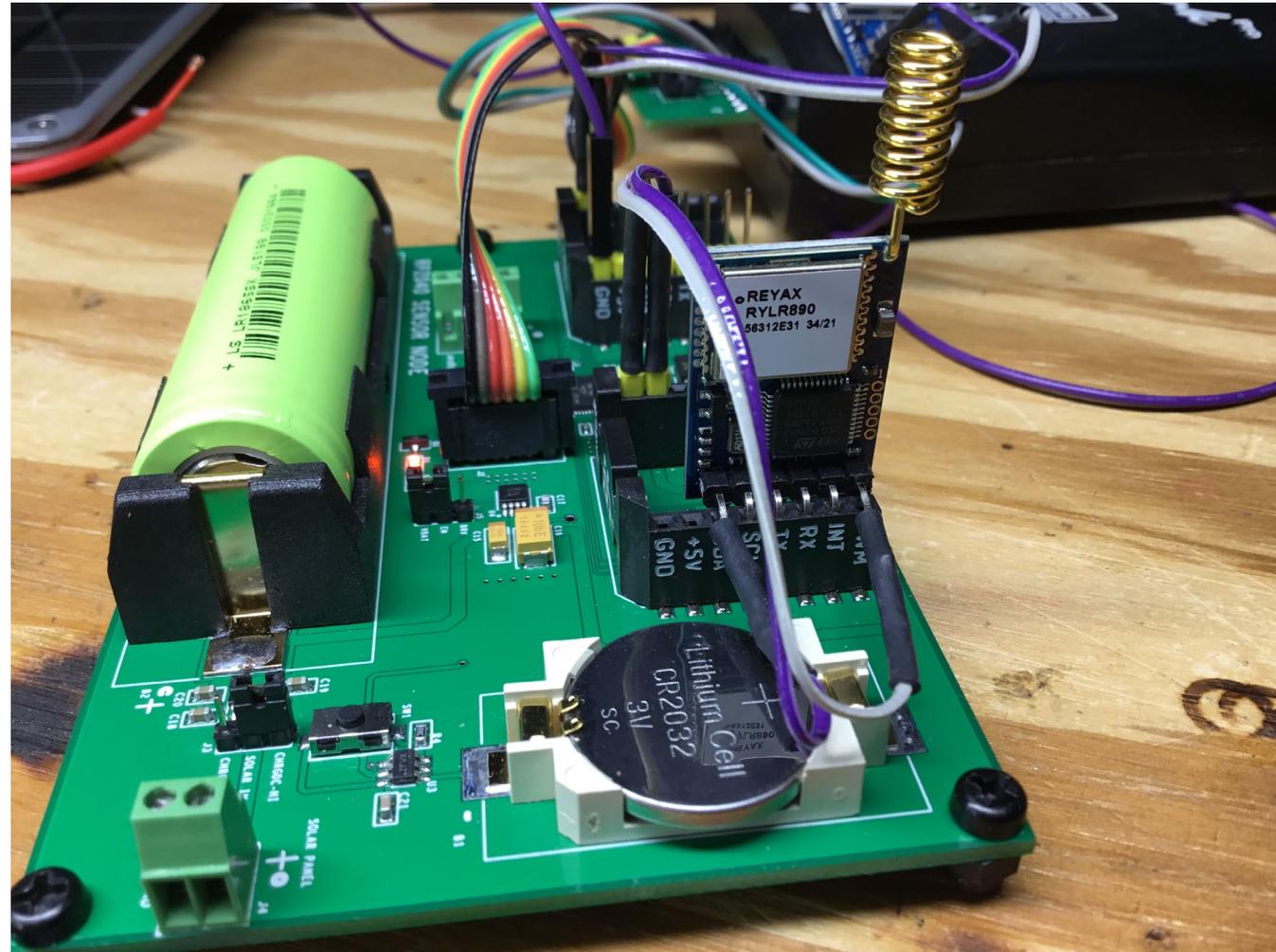
2B 52 45 41 44 59 0D 0A
2B 4F 4B 0D 0A
4B

```

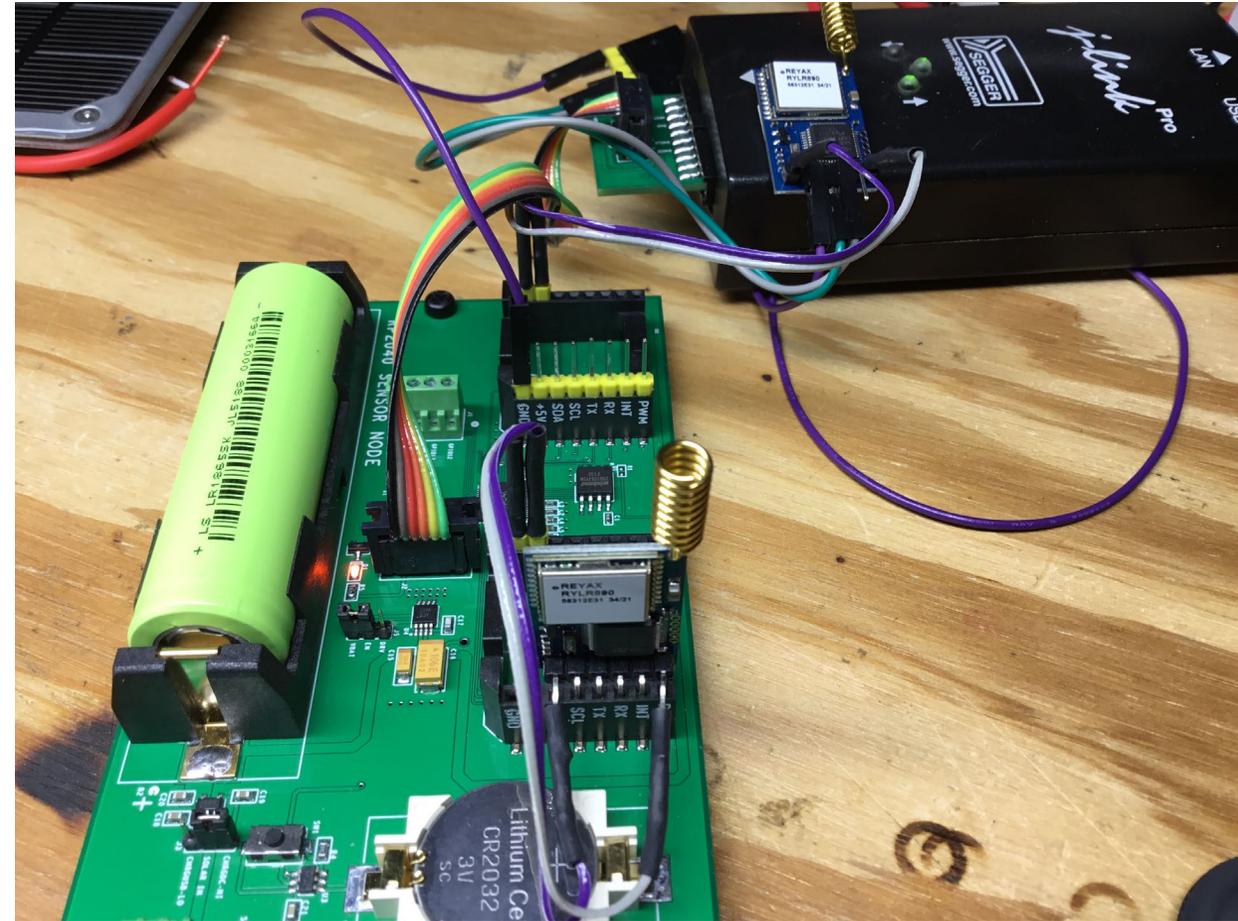
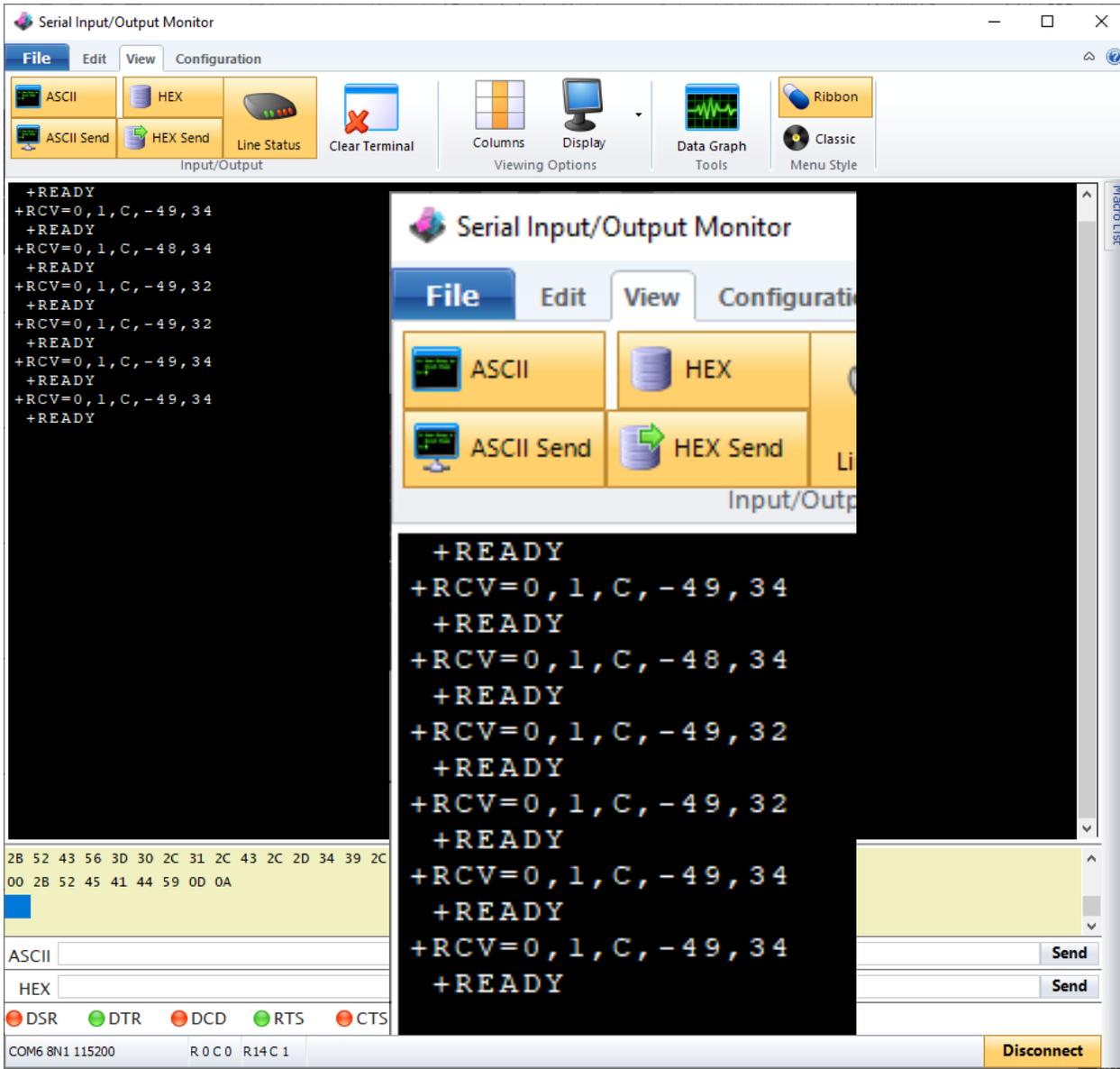
At the bottom of the application, there is a status bar showing the current port configuration: COM6 8N1 115200, R 0 C 0 R 4 C 2, and a Disconnect button.

REYAX RYLR896 - sendMsg

```
117     case sendMsg:
118
119         uart_puts(uart1, "AT+SEND=0,1,C\r\n");
120         // DEBUG MONITOR START
121         gpio_put(LED_PIN, true);
122         // DEBUG MONITOR END
123         sleep_ms(900);
124         gpio_put(DONE_PIN, true);
125         while(1);
126     break;
127 }
128 }
129 }
130 }
```



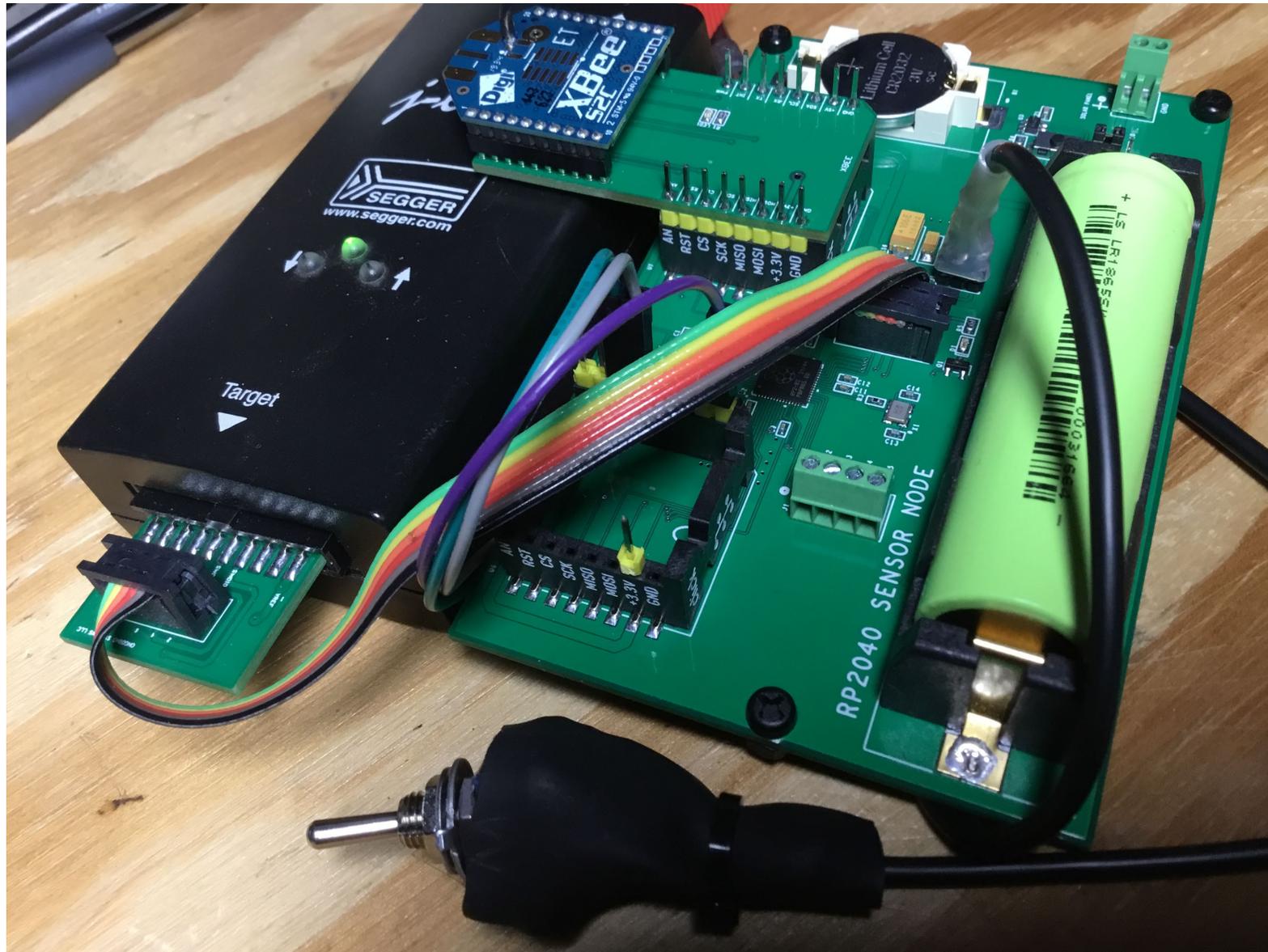
Low-Power REYAX RYLR896 Communications



The Fleet



Run/Low-Power Test Setup

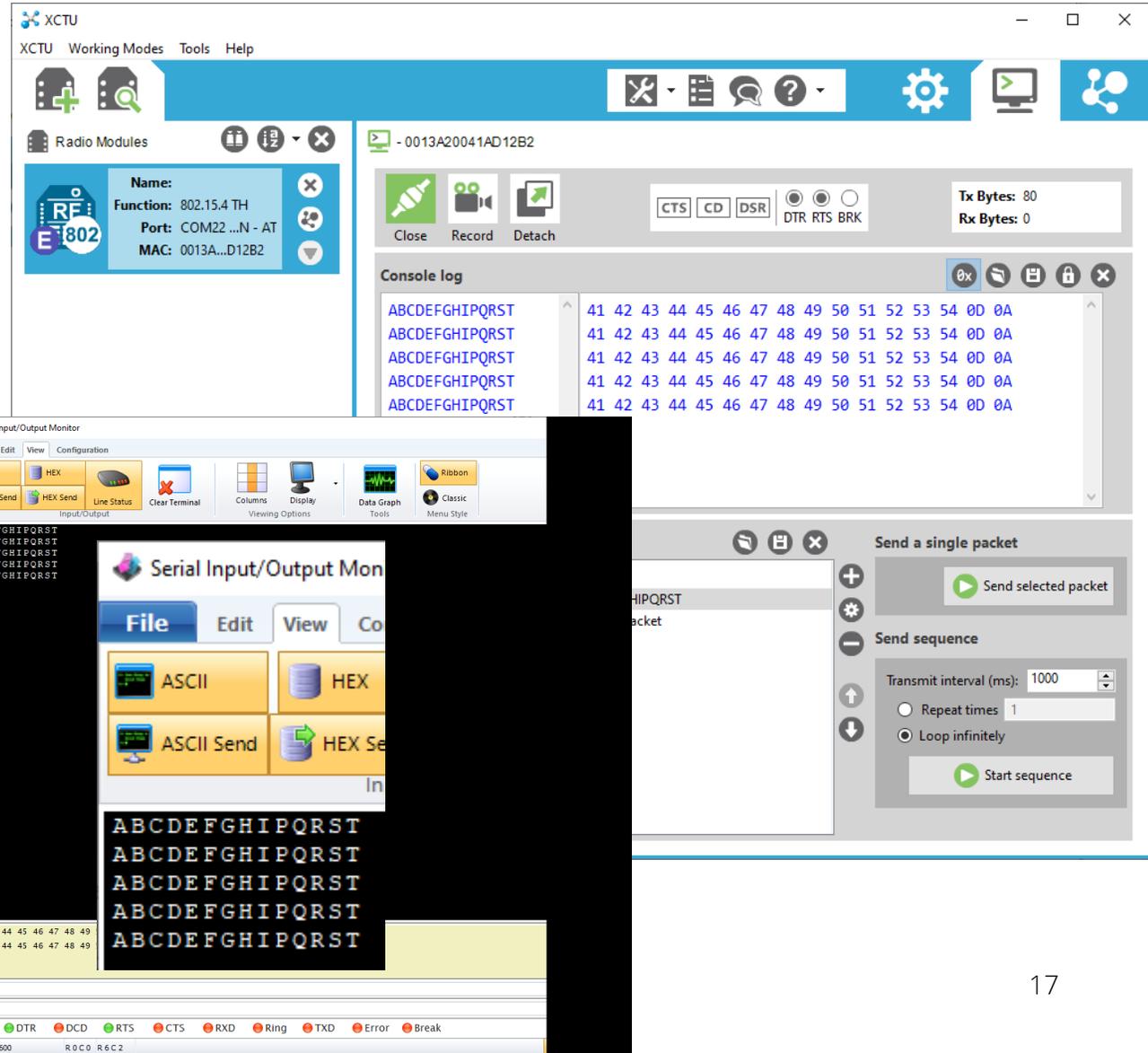


UART1 Receive Interrupt Test

```

69 // Init state machine variable and init RX buffer index
70 state = wait4ready;
71 indx_1 = 0;
72 // Pull XBee out of reset
73 sleep_ms(1000);
74 gpio_put(NRST1_PIN, true);
75 while (1)
76 {
77     switch(state)
78     {
79         // Look for traffic from the XBee
80         case wait4ready:
81             if(indx_1 > 0)
82             {
83                 sleep_ms(100);
84                 // DEBUG MONITOR
85                 for(i=0;i<indx_1;i++)
86                 {
87                     uart_putc(uart0, rxBuf_1[i]);
88                 }
89                 // DEBUG MONITOR END
90                 indx_1 = 0;
91                 //sleep_ms(1000);
92                 //uart_puts(uart1, "SENSOR DATA\r\n");
93                 //sleep_ms(1000);
94                 //gpio_put(DONE_PIN, true);
95                 //while(1);
96             }
97             break;

```



The image shows two software interfaces used for testing the UART1 receive interrupt on a Raspberry Pi RP2040.

XCTU (Xbee Configuration Tool Utility): The top window shows the configuration for an XBee module. The Name is "0013A20041AD12B2". The Function is "802.15.4 TH". The Port is "COM22 ...N - AT". The MAC is "0013A...D12B2". The console log shows the received data: "ABCDEFHGHIQRST" repeated five times, with the corresponding hex values: "41 42 43 44 45 46 47 48 49 50 51 52 53 54 0D 0A".

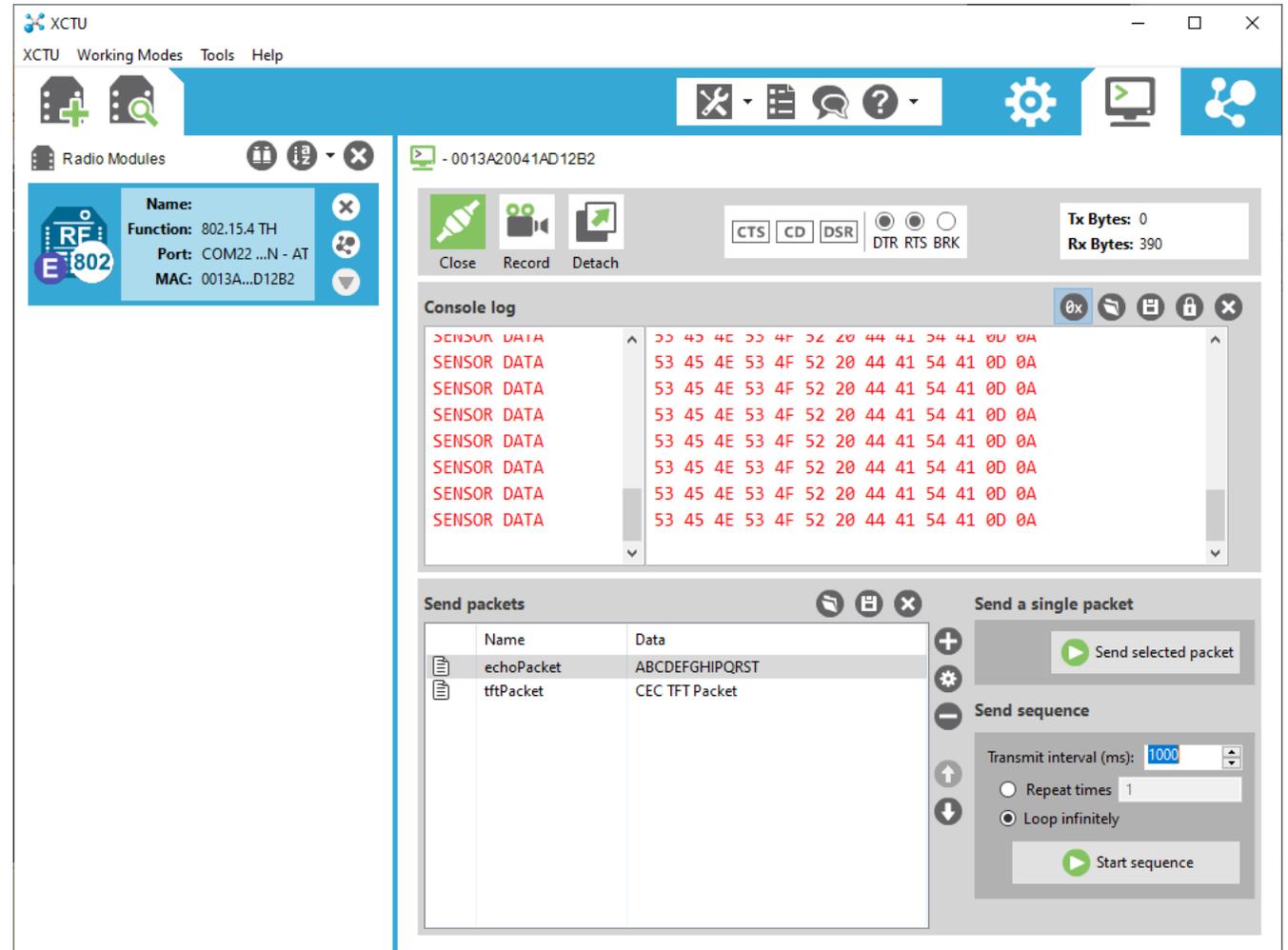
Serial Input/Output Monitor: The bottom window shows the serial data being received. The data is displayed in ASCII and HEX. The ASCII data is "ABCDEFHGHIQRST" repeated five times. The HEX data is "41 42 43 44 45 46 47 48 49 50 51 52 53 54 0D 0A" repeated five times. The status bar at the bottom shows the port as "COM6 9611 9600" and the baud rate as "R0C0 R6C2".

Low-Power XBee Communications

```

69 // Init state machine variable and init RX buffer index
70 state = wait4ready;
71 indx_1 = 0;
72 // Pull XBee out of reset
73 sleep_ms(1000);
74 gpio_put(NRST1_PIN, true);
75 while (1)
76 {
77     switch(state)
78     {
79         // Look for traffic from the XBee
80         case wait4ready:
81             //if(indx_1 > 0)
82             //{
83                 //    sleep_ms(100);
84                 //    // DEBUG MONITOR
85                 //    for(i=0;i<indx_1;i++)
86                 //    {
87                     //        uart_putc(uart0, rxBuf_1[i]);
88                 //    }
89                 //    // DEBUG MONITOR END
90                 //    indx_1 = 0;
91                 sleep_ms(1000);
92                 uart_puts(uart1, "SENSOR DATA\r\n");
93                 sleep_ms(1000);
94                 gpio_put(DONE_PIN, true);
95                 while(1);
96             }
97             break;

```



The screenshot shows the XCTU software interface. On the left, a radio module is configured with the following details:

- Name: 802.15.4 TH
- Function: 802.15.4 TH
- Port: COM22...N - AT
- MAC: 0013A...D12B2

The console log displays the following sensor data:

```

SENSOR DATA  >> 43 4E 53 4F 52 20 44 41 54 41 0D 0A
SENSOR DATA  53 45 4E 53 4F 52 20 44 41 54 41 0D 0A
SENSOR DATA  53 45 4E 53 4F 52 20 44 41 54 41 0D 0A
SENSOR DATA  53 45 4E 53 4F 52 20 44 41 54 41 0D 0A
SENSOR DATA  53 45 4E 53 4F 52 20 44 41 54 41 0D 0A
SENSOR DATA  53 45 4E 53 4F 52 20 44 41 54 41 0D 0A
SENSOR DATA  53 45 4E 53 4F 52 20 44 41 54 41 0D 0A

```

The 'Send packets' section shows a table of packets:

Name	Data
echoPacket	ABCDEFGHPIQRST
tftPacket	CEC TFT Packet

The 'Send a single packet' section has a 'Send selected packet' button. The 'Send sequence' section has a 'Transmit interval (ms): 1000' field, radio buttons for 'Repeat times 1' and 'Loop infinitely', and a 'Start sequence' button.

Thank you for attending!!!

Please consider the resources below:

- raspberrypi.org
- [RP2040 Data Sheet](#)
- www.digi.com
- www.reyax.com

MORE TO COME..





DesignNews

Thank You

Sponsored by

