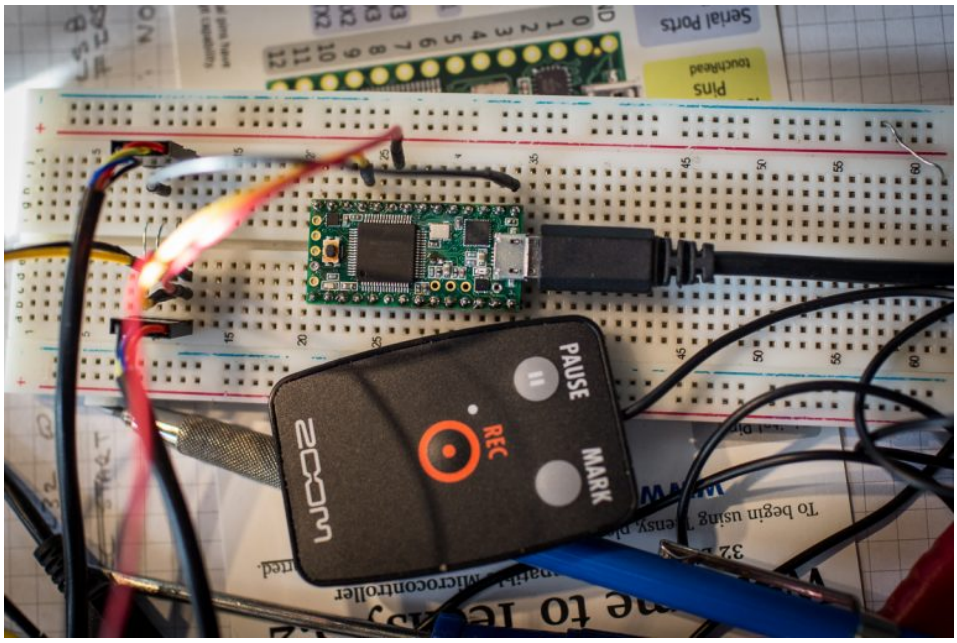
ARDUINO, ELECTRONICS, HARDWARE, INTERESTING

CONTROLLING THE ZOOM H2N AUDIO RECORDER WITH ARDUINO

17TH APRIL 2017 | KARL

In Part One I covered the byte codes sent by the Zoom Remote Controller RC1 and decoded data sent over the wire to the remote from the Zoom H2n Recorder.

In this post I will be covering the use of an Arduino style micro-controller to decode the signals sent by the remote, then control the recorder. I have used a Tennsy 3.1 Arduino clone as this is a small controller with two additional hardware serial ports, works with 3.3volt logic, and a with the addition of a crystal and button battery a real-time clock.



Arduino control of the Zoom H2n

Setup and Connections

The connections on the remotes four pin 2.5mm jack, with pin one being the tip:

1. Remote Receive – RX
2. Remote Transmit – TX
3. Ground

4. 3.1V – Power

On the Teensy there are two hardware serial UARTs available in addition to that used by the USB port, UART2: Pin 9 (RX2), Pin 10 (TX2) and UART3: Pin 7 (RX3), Pin 8 (TX3). Serial data is sent at 2400 baud, 8 bits, no parity, 1 stop (8n1). The response data shown is for when the recorder is in XY Stereo mode (0x20, 0x21), different codes are returned when other recording modes are used, see the end of Part One for details.

Serial Monitor

This first chunk of code is for monitoring the outputs of the remote control and recorder. Connect Remote Receive – RX on the remote to RX2 – Pin 9 on the Teensy and Remote Transmit – TX to RX3 – Pin 7 and Ground to Ground on the Teensy. This program will output data received to the Arduino IDE's serial monitor.

```
1  //written for a Teensy 3.x
2  #define MONITOR_LED 13
3  int incomingByte2 = 0;
4  int incomingByte3 = 0;
5  unsigned long nowMillis = 0;
6
7  void printByte(int p, int b) {
8    char out[40];
9    sprintf(out, "RX%d: %8lu\t0x%x\t%d",p,nowMillis,b,b);
10   Serial.println(out);
11 }
12
13 void setup() {
14   delay(1000);
15   Serial.begin(9600);
16   Serial.println("ready");
17
18   // incoming data
19   Serial2.begin(2400, SERIAL_8N1);  // Remote Receive - RX
20   Serial3.begin(2400, SERIAL_8N1);  // Remote Transmit - TX
21
22   pinMode(MONITOR_LED, OUTPUT);
23   digitalWrite(MONITOR_LED, HIGH);
24 }
25
26 void loop() {
27
```

The output is in four columns; the UART seeing activity, current milliseconds and the received data in hexadecimal and decimal values.

Taking Control – But Not Listening

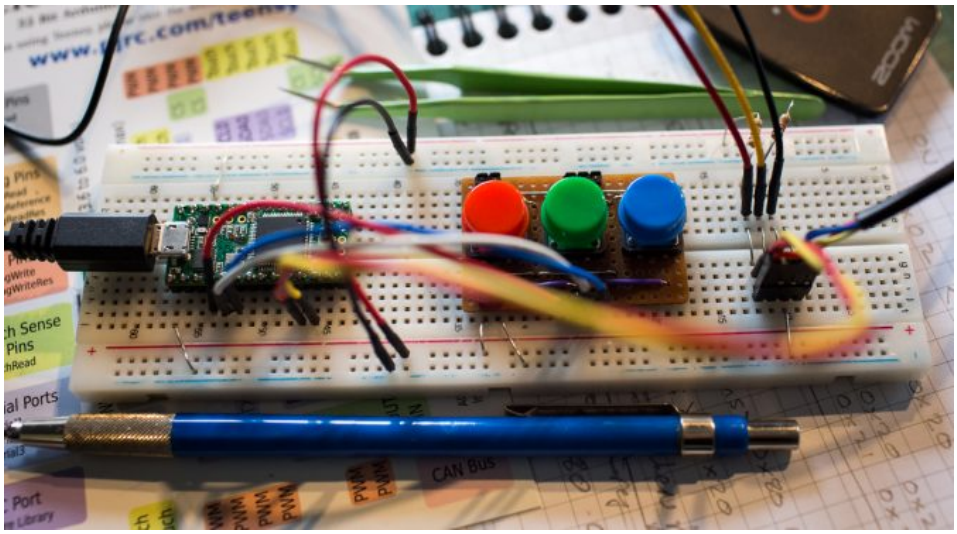
Sending the command to the recorder blindly is quite straight forward, just send the bytes to the Remote Transmit – TX pin on the recorder (Zoom RX). This can be seen in the following, when run it starts the recorder recording for ten seconds. Connect: Remote TX to TX2 on the Teensy, Remote RX to RX2 and Ground to Ground.

```
1  //written for a Teensy 3.x
2  #define MONITOR_LED 13
```

```
3
4  // bytes to start and stop the recorder, a negative number is used as a delay
5  int record[5] = { 0x81, 0x0, -100, 0x80, 0x0 };
6
7  void zoomTX(int d[], int len) {
8
9      // show what is going to be done
10     for (int i = 0; i < len; i++) {
11       char out[40];
12       if (d[i] < 0) {
13         sprintf(out,"TX2: delay(%d)",abs(d[i]));
14         Serial.println(out);
15       }
16       else {
17         sprintf(out, "TX2: %d\t0x%x", i, d[i]);
18         Serial.println(out);
19       }
20     }
21
22     // do the command
23     for (int i = 0; i < len; i++) {
24       if (d[i] < 0) {
25         delay(abs(d[i]));
26       }
27       else {
```

Taking Control – And Listening for a Reply

The next stage is to have the Teensy control the Zoom and listen for a response from the recorder. Again, as before connect: Remote TX to TX2 on the Teensy, Remote RX to RX2 and Ground to Ground. For the demonstration I have added three buttons to act as the controller.



Zoom Control with three buttons

The following code needs more development work, I ran out of time, but I think gives a good starting point for further investigation. I have placed the commands for the remote in a structure, each command; record, pause and mark has four components, the command to transmit to the Zoom, the expected responses when starting and stopping and a flag to store the status.

```
1  //written for a Teensy 3.x
2
3  #define MONITOR_LED 13
4  #define RED_BTN 2
5  #define GRN_BTN 3
```

```
#define BLU_BTN 4

unsigned long nowMillis = 0;
unsigned long prevMillis = 0;
unsigned long buttonPressDelay = 400;
unsigned long serialTimeout = 8000; // it can take a few seconds for the zoom to start recor

struct ZOOMTX {
  int record[4][5] = { {0x81, 0x0, -100, 0x80, 0x0},   // Transmit command data - a minus nu
                       {0x20, 0x20, 0x21},             // Expected Response - Start (in XY S
                       {0x21, 0x21, 0x20},             // Response - Stop
                       {0} };                          // status: record[3][0] = 1: recordir
  int pause[4][5]  = { {0x80, 0x2, -100, 0x80, 0x0},
                       {0x21, 0x21, 0x20},
                       {0x20, 0x21},
                       {0} };
  int mark[4][5]   = { {0x80, 0x1, -100, 0x80, 0x0},
                       {0x20, 0x20},
                       {0x20, 0x20},
                       {0} };

```

There is a problem when resuming from pause, because the Zoom sends codes to flash the LED on the remote this can pick up the wrong pair of bytes; such as 0x21 0x21 instead of the expected 0x20 0x21.

I expect to be revisiting this, adding a timer function plus external battery for long running. I'm not sure how useful listening for a response is, sending the record command toggle on its own seems fairly robust without the need to check.

Links and Sources

- Teensy UART details
- DIY Remote Control for Zoom H4n and Canon 60D