Set up Arduino UNO for CW via midi

Connecting a key or keyer to an SDR console like Quisk, SparkSDR, HPSDR variants is difficult over an ordinary serial input due to the long latency in the order of a dot length at 25 wpm and by the time the tone from the keyer paddle press is received from the radio sidetone the second dot is on the way. At 15 wpm the lag is acceptable but still on the verge of sending the next code element before the ear and fingers can respond to the end of the first. The normal solution is to plug the key/ keyer directly into the Hermes-Lite and have a sidetone produced from there rather than the radio. The recent advances in remote operation of the station makes this difficult to do as remote cabling or wireless link etc. are needed to bring the key to the remote site. Enter MIDI.

MIDI is a low latency protocol for the interconnection of musical equipment and has been developed for the lowest latency that our current technology can produce. It can provide output as a TTL serial output with a standard 5 pin DIN connector or as USB in a standard format recognised by the main platforms and not requiring a special driver similar in nature to HID devices like mouse and keyboard. It seems that the latency in our SDR environments is in the order of 12 mSec and there would not be many operators in the world that can out-send that.

There are a number of Arduino devices like the Pro micro and teensy that understand the midi usb protocol natively and can be pressed into service easily but the Arduino UNO and the Mega can also be arranged to run in MIDI USB mode as they have a small microprocessor (ATMEGA16u2) separate from the main microprocessor to do the TTL to USB serial connection to a host computer and this can be reprogrammed to perform its original function or become a midiUSB port.

Programming the ATMEGA16U2 microprocessor

The UNO 16u2 microprocessor is programmed with the dualMoco.hex file available from:
https://github.com/kuwatay/mocolufa/blob/master/HEX/dualMoco.hex
There does not seem to be a download for the .hex file so it is best to copy all the text and paste into a text editor like xed and save the file with the name dualMoco.hex in a convenient location.
The next step is to burn the hex file and this is done with dfu-programmer. Check to see if this is already installed with from a terminal

```
dfu-programmer – version
```

If not installed, install with

```
sudo apt install dfu-programmer
```

Navigate in a terminal to where the dualMoco.hex file is installed and plug the Arduino UNO via a USB cable into the computer. Run the following commands:

```
sudo dfu-programmer atmega16u2 erase
sudo dfu-programmer atmega16u2 flash dualMoco.hex
sudo dfu-programmer atmega16u2 reset
```

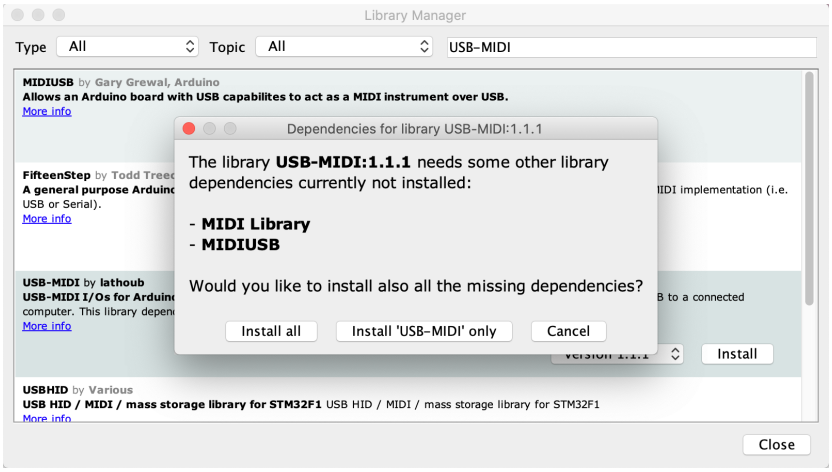
Check to see if there is a midi device installed

```
lsusb
Bus 004 Device 002: ID 0718:1906 Imation Corp.
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 009: ID 03eb:2048 Atmel Corp. LUFA MIDI Demo Application
Bus 003 Device 003: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

The LUFA MIDI device should be showing.

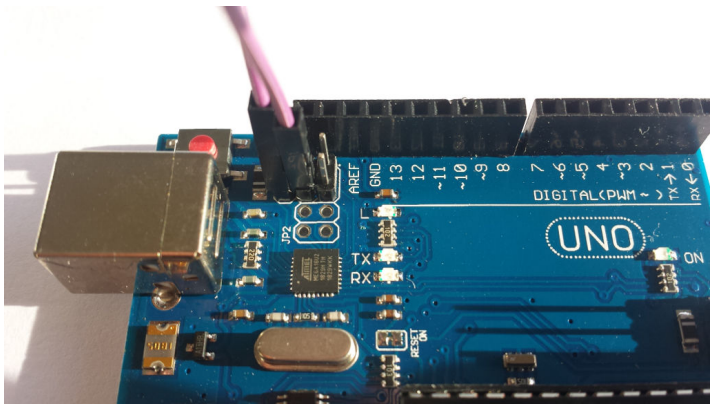
Programing the Arduino UNO with the CW sketch

The first step is to install the required libraries on the Arduino IDE.



The new USB midi interface needs a library which is available on the Arduino IDE and can be installed via the Tools/Manage Libraries (Ctrl+shift+I) and type USB-MIDI into the filter box. The library required is USB-MIDI by lathoub and the current version is 1.1.2 it will ask for dependencies and you should choose Install All.

Next step is to load a sketch into the Arduino UNO ATMEGA328p processor as follows:



Unplug the Arduino USB cable from the computer and place a jumper between pins 4 & 6 on the 16u2 isp header to switch the UNO into serial mode to allow it to be programmed from the Arduino IDE in the normal way.

I wrote a short sketch to accept input on the same pins as Alan and this should be set up on the Arduino IDE and is reproduced below:

```
#include <MIDI.h> // Include MIDI Library
MIDI_CREATE_DEFAULT_INSTANCE(); // Create an instance of the midi library
#define C3 48  // Define some notes

const int BUTTON_PIN_COUNT = 2;

// Change the order of the pins to change the ctrl or note order.
int buttonPins[BUTTON_PIN_COUNT] = { 2, 3 };

int buttonDown[BUTTON_PIN_COUNT];



int isButtonDown(int pin) {
  return digitalRead(pin) == 0;
}
```

```
void setup() {
  //  MIDI.begin(MIDI_CHANNEL_OMNI); // Begin MIDI and listen to all channels
  MIDI.begin(1); // Begin MIDI
  for (int i = 0; i < BUTTON_PIN_COUNT; ++i) {
    pinMode(buttonPins[i], INPUT);
    digitalWrite(buttonPins[i], HIGH);
    buttonDown[i] = isButtonDown(buttonPins[i]);
  }
}

void loop() {
  for (int i = 0; i < BUTTON_PIN_COUNT; ++i) {
    int down = isButtonDown(buttonPins[i]);

    if (down != buttonDown[i]) { // See if key state has changed
      if (down) {
        MIDI.sendNoteOn(64+i, 127, 1);
      }
      else {
        MIDI.sendNoteOff(64+i, 0, 1);
      }
    }
    buttonDown[i] = down;
  }
}
```

It needs a bit of refining but will do for testing. To load it into the UNO, fit the jumper onto the 16u2 header and plug in the USB cable. You will need to press the reset button on the UNO and sometimes I needed to do it more than once but the port should eventually show.

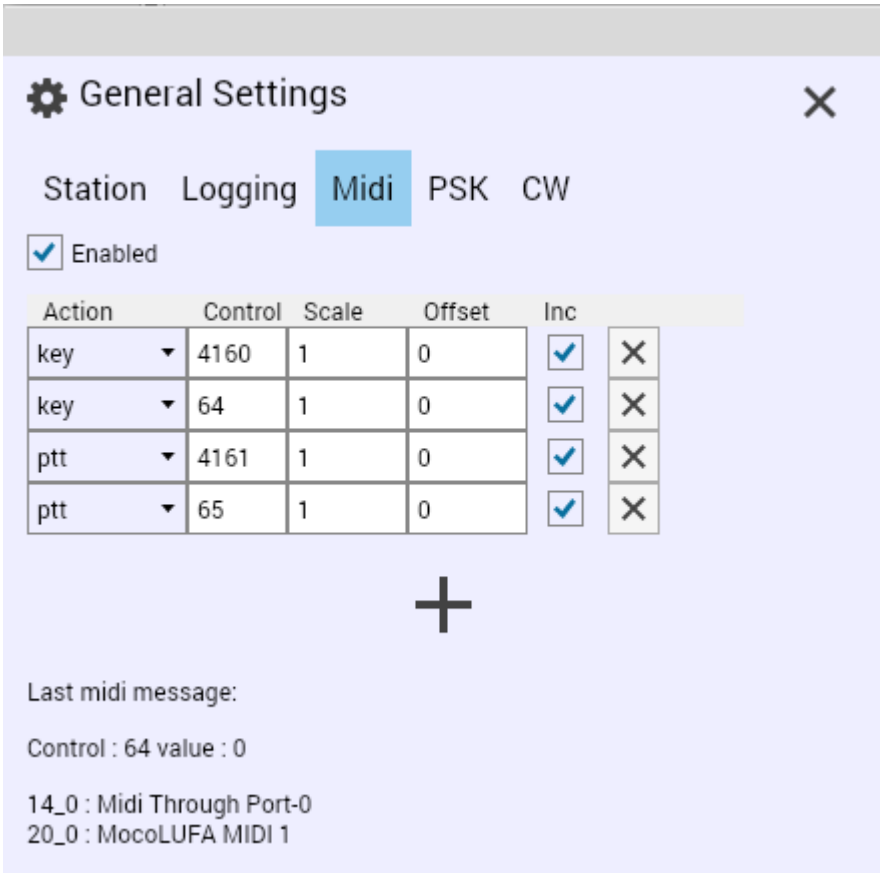
Set your usual programming settings and upload the sketch.
Unplug the USB cable and remove the jumper on the 16u2 header
Plug the USB cable back in and open a terminal and check as follows

```
gvj@gvj-M92p ~ $ aseqdump -l
Port Client name Port name
0:0 System Timer
0:1 System Announce
14:0 Midi Through Midi Through Port-0
20:0 MocoLUFA MocoLUFA MIDI 1

From the above, I can see my midi device (MocoLUFA) and using sketch to send
data I touched a wire from D2 to ground on the Uno to send a midi signal.

gvj@gvj-M92p ~ $ aseqdump -p "MocoLUFA"
Waiting for data. Press Ctrl+C to end.
Source  Event                  Ch  Data
 20:0   Note on                 0, note 64, velocity 127
 20:0   Note on                 0, note 64, velocity 127
 20:0   Note off                0, note 64, velocity 0
 20:0   Note on                 0, note 64, velocity 127
 20:0   Note off                0, note 64, velocity 0
^C
gvj@gvj-M92p ~ $
```

Setting up SparkSDR

Start up SparkSDR and get a radio running.

1. Choose the settings menu and under General Settings/Station/Advanced, check the “Enable Experimental Features” checkbox.
2. Click on the “Midi” panel and check the “Enabled” checkbox
3. The “MocoLUFA MIDI 1” connection should be visible.
4. Jumper D2 on the UNO to ground and note the Control: value. Mine is 4160 value:127
5. Remove the jumper and note the Control : value. Again mine is 64 value:0
6. Click the + (Add new command) and a command entry line will appear.
7. Choose the action which I chose “key” for
8. Enter one of your control values and leave the scale and offset at the defaults.
9. Add another entry line and fill on the other set of control values.
10. Finally repeat the steps for the control signals from D2 on the UNO only this time using D3 and for the action I used “ptt”.



The input on D3 seems to be a toggle i.e. press on/press off and needs some key debouncing so more to come.

Graeme ZL2APV