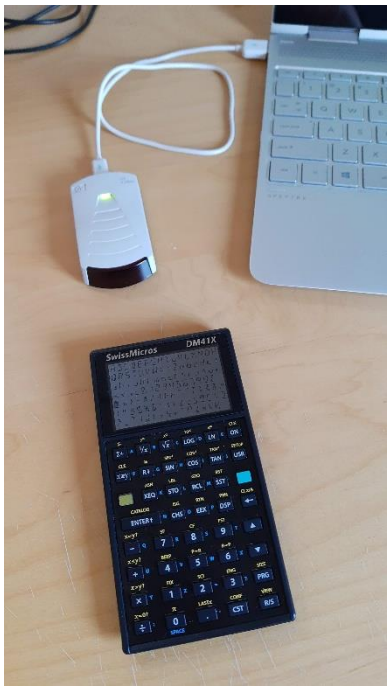
HP-IR Receiver with Arduino



Meindert Kuipers – April 2020, updated December 2022

After getting my DM41X and playing with it I realized that it did not have many options to exchange data with the outside world, unlike the real HP41. The only possibility appeared to be the use of the built-in USB disk, accessible form a connected computer. After further reading the documentation I realized that it had an IR transmitted that could work with IR enabled printers, mainly the HP 82240B printer. After some searching I then found an article by Martin Hepperle that used an off-the-shelf HP IR transceiver (used for the HP Laserprinters), that was modified to act as an IR receiver for a broad range of HP calculators, and could interface to a PC with a printer simulator. The complete article can be found at https://www.mh-aerotools.de/hp/red-eye/index.htm , with the sources at https://www.mh-aerotools.de/hp/red-eye/IR_RedEye-Arduino-1.0.5r2.zip . This implementation worked almost completely with very few modifications. This article explains how I implemented the RedEye receiver. Please make certain that you understand Martins work before trying to implement this yourself.

Many thanks Martin, for an excellent piece of work!

Hardware needed

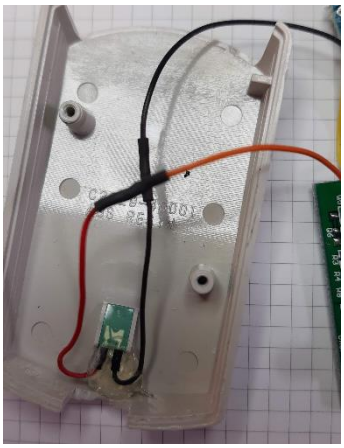
First of all, get your hands on the HP IR transceiver, part number C4103A. It is available at eBay for a few Euro's, depending on the seller. The version that I bought looks a bit different from the one in Martins article, the functionality and pinout is identical.

Other important part is the Arduino. I have decided to use a Pro Micro Arduino. This is not an official Arduino product, but a rather smart version designed by SparkFun (see https://www.sparkfun.com/products/12640 , and make sure you use the 5V version). I actually bought a clone of that design. The advantage of the Pro Micro is that is has an integrated USB to serial interface, and it is small enough to be built inside the IR Transceiver housing. There are no buttons, so reset and upload are completely done from the Arduino IDE. This was actually my first ever Arduino project!

For the rest you only need a few wires and a 10k resistor, plus some (soldering) tools and skills of course.

IR Receiver

Open the IR receiver by removing the label on the bottom, and take out the screws (mine had only two screws). It should come apart quite easy. I did a first test with a breadboard, but if you are brave enough you can start with removing the cable by pulling the wires from the connector. Else just connect the Arduino board with wires to the DIN connector and make the connections as described later to ensure that everything works.



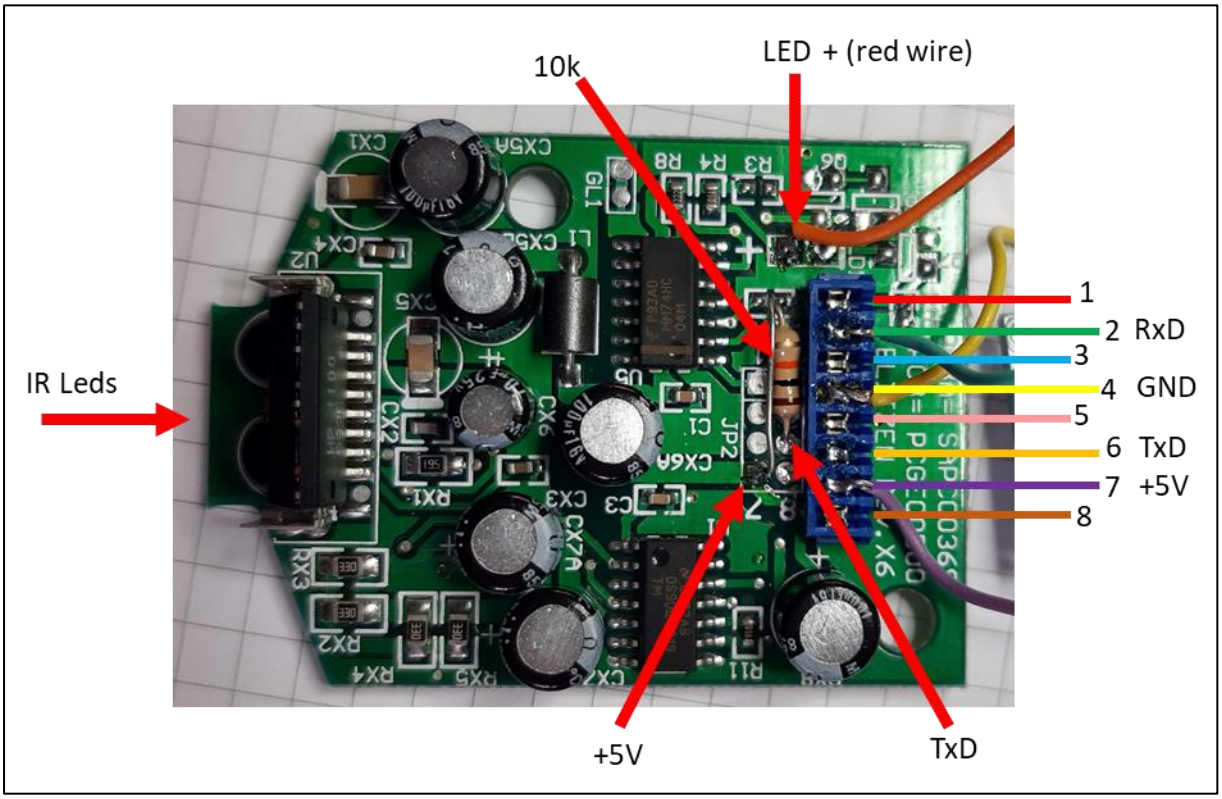
The IR Receiver has an LED in the housing, and to make some good use of it (I still have to modify the software, I only have added a blink at initialisation) it is best to connect this to the Arduino so it can be controlled from your own software. I have unsoldered both wires to the LED on the PCB (to make them a bit longer for easier handling), but you can still leave the red wire in place. It is connected to Vcc through the smd resistor R4, so why not use that. The black wire will be connected to the Arduino pin D5.

I do not intend to implement any IR transmitting functions, so I used a 10k resistor to tie the TxD signal high (to prevent that the IR transmit LED is constantly on). This can be done best using the pinout for a header (I realized too late that the other wires could use this as well …).

In my setup only 3 wires are used from the connector on the IR receiver (pins counted from the top in the picture)

- Pin 2 – RxD, green wire
- Pin 4 – GND, yellow wire
- Pin 6 – TxD, orange wire, tie to Vcc with a 10-20k pullup, optionally connect to an Arduino pin and drive high, or actually use this to send data
- Pin 7 – Vcc +5V, purple wire

I have soldered the wires in the connector, with a fine pin of the soldering iron. The wires could be connected on the header pads, and also on the bottom. I did not use the bottom pins of the connector as there is not much room between the PCB and housing bottom.



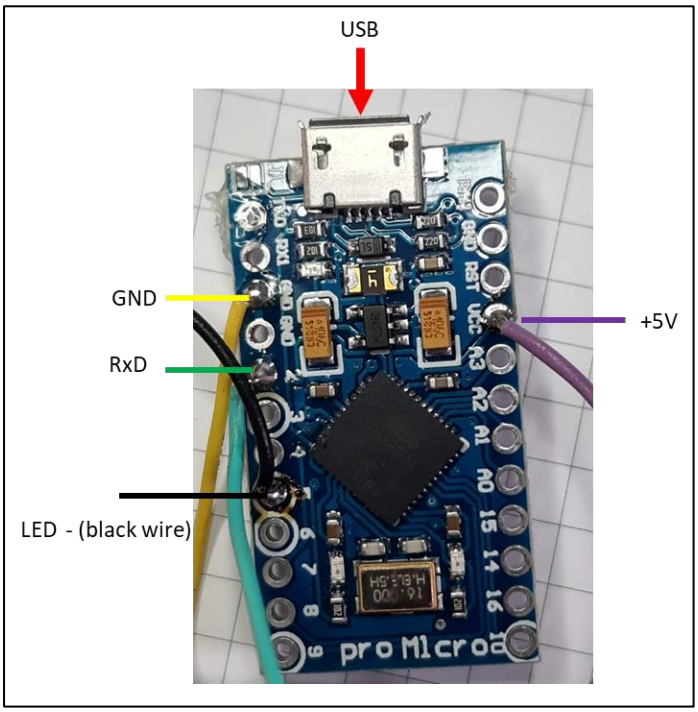
Arduino Pro Micro

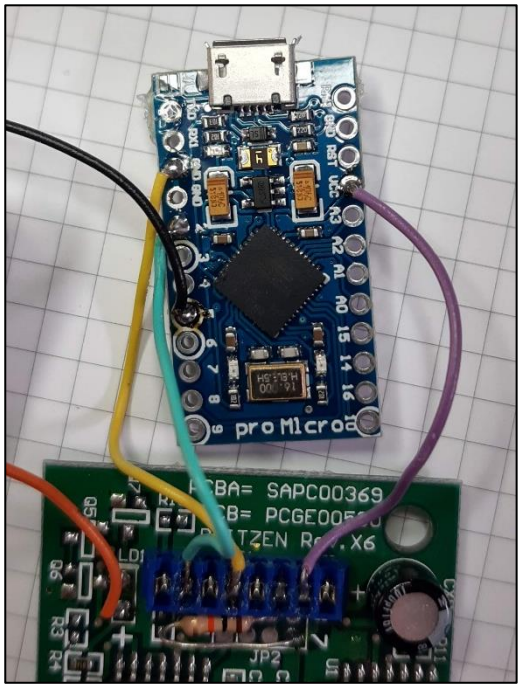
The connections on the Arduino are pretty simple. GND and +5V Vcc connect to the IR receiver, this powers the receiver from USB.

The RxD signal is connected to pin 2 (D2 of the microcontroller) on the Pro Micro, this matches with INT1 in the original software. On the Arduino Pro Mini used in Martin Hepperle's article this is pin 3! With this setup Martins software worked immediately without any modifications.

I have connected the black LED wire (the LED in the receiver housing) to D5, this pin can also be used for PWM output, to bring some nice effects to the LED.

Sparkfun has a really helpful Hookup Guide for the Pro Micro on
https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide

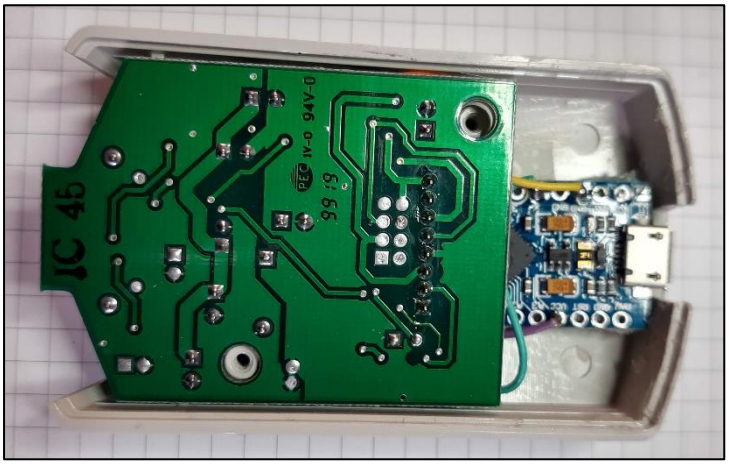


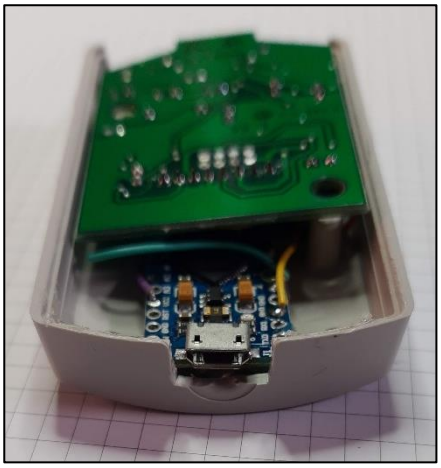


Assembly

First make the cable hole in the receiver top housing a bit larger so the USB connector can be accessed easily by the cable. I made mine a bit too large, but it works. You may have to do the same on the housing bottom. I fixed the Arduino board with some hot glue to the top housing. It will be at a slight angle to allow enough space for the connector on the receiver board, but everything fits nicely. Finally the housing can be closed.

Of course first test if everything works before applying the glue and closing the housing. Especially verify if the pullup resistor is connected by looking at the transmitter LED with your phone or digital camera. According to Martin's article it can become hot if it transits all the time. When on it can be easily seen with phone or camera.





Afterword

This was my first Arduino project, and it turned out to be surprisingly easy. I tested my first Blinky program (of course) with the Arduino Pro, the second project was already the RedEye receiver. Apart from first setting the hardware to the correct Arduino hardware, it was really straightforward. So far the only modification I made was to add a welcome blink for the led in the receiver housing, and a heartbeat by using the PWM. I have not changed the baud rate, as the DM41X that I am using it with is sending data at a very low speed.

Also the hookup to the PC was easy, you may have to look for the correct COM port, but you can send printout to a serial terminal emulator or printer emulator. When using a terminal emulator, press T to put the software in Translate mode. Depending on the terminal program translation of LF to CR/LF may be needed.

This RedEye receiver has been tested with the DM41X, a WP34S and I have seen reports from users using it with the DM42. When using with the DM41X, it is recommended to set the Printer Line Delay (via Settings) to the lowest possible value. When using an emulated printer there is no need to have a delay to compensate for the printer head speed.

If you build it: have fun!