Eric Donders 2020-11-20

RemoteControl Documentation

Summary

The RemoteControl program running on a Teensy 2.0 microcontroller allows the Nintendo Switch to be controlled by a computer. To do so, the Teensy acts as an intermediary that converts signals from the computer, in the form of characters sent by serial communication, to controller inputs. This document describes the physical configuration of the intermediary microcontroller and provides instructions for sending commands to it from any computer equipped with a USB virtual COM port.

Note that this program is designed primarily for automation purposes. No effort has been made to optimize response time and thus real-time gaming, especially where fast reactions are needed, is not recommended.

Before you start

This project is based on the AutoController program family developed by brianuuuuSonic. Please download his programs and watch his video on the basic hardware you will need:

https://www.youtube.com/watch?v=y2xFf7e_KSU&list=PLrAfKLfOSiGFv6wjohlEXQqYjEIoBOkfX

Additional hardware

This project requires some additional hardware that allows the Teensy 2.0 and the computer to communicate. I used a breakout board that houses a USB to serial converter which is produced by Adafruit:

https://www.adafruit.com/product/3309

Alternatively, you can purchase a USB to serial cable that packages this hardware into the USB hub itself.

Regardless of how you approach the communication between the Teensy and the computer, you will likely have to do some soldering since the Teensy's second hardware serial port is only accessible through pins PD2 and PD3 on the left side of the Teensy. Figure 1 describes how to connect the Teensy and USB to serial converter to enable two-way communication (TXD and RXD) and ensure the devices share a common ground voltage (GND).

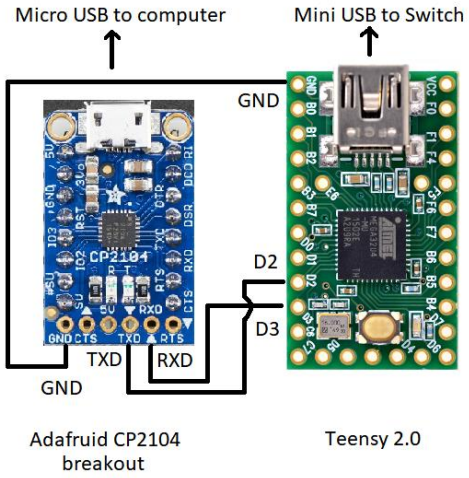

Figure 1: Wiring diagram describing how to attach the USB to serial converter to the Teensy 2.0. Note how the TDX (transmit) pin of the CP2014 is connected to pin D2 (RXD1, receive) of the Teensy and vice versa. Although this configuration might seem unintuitive at first, it makes sense when you consider that each device must transmit its data to the other's receiving pin.

Sending commands to the microcontroller

The program accepts commands in the form of single characters sent by serial communication at 9600 Baud. It will echo back any character that you send it as a debugging feature; if you do not receive a response to a sent character then you know what something is wrong with the program or with the wiring. You can use a serial monitor such as the one included in the Arduino IDE to verify that everything is working before moving on to a more complex controller program. Figure 2 shows the characters corresponding to each controller button or joystick input.

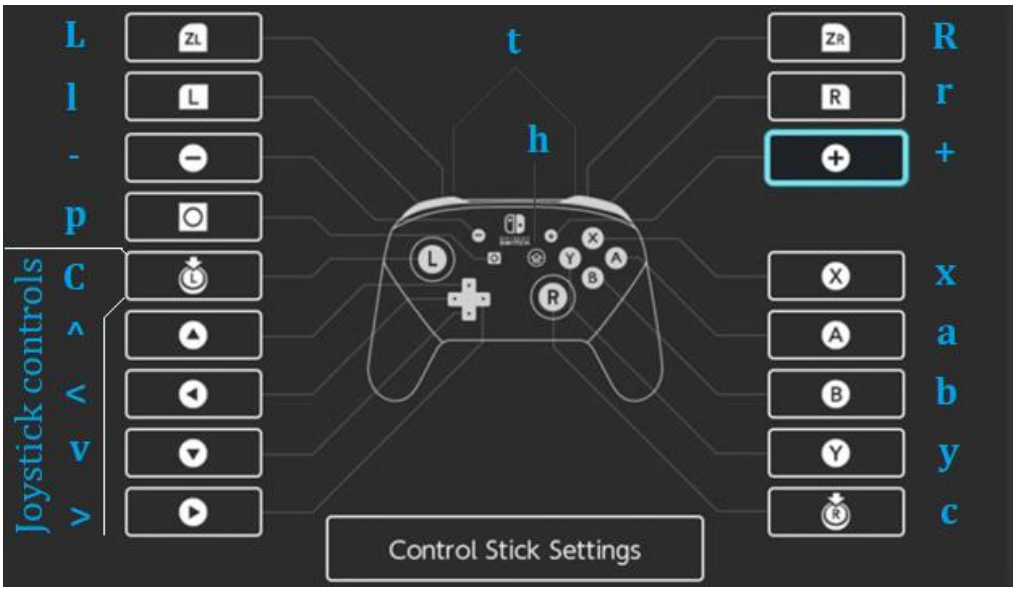

Figure 2: Mapping of commands and controller buttons on the Switch. These settings, along with the Baud rate, can be changed by modifying the source code and recompiling the .hex file. Image adapted from https://daniele-michelin.com/index.php/2020/04/19/heres-how-to-remap-the-buttons-on-your-nintendo-switch-controller/.

Compiling the source code

The code for this program was modified from the “TurboA” program developed by brianuuuSonic and shares many of the same dependencies. To compile the code, I made a new folder called “RemoteControl” for this program under the …/Sourcecode/Bots/ folder and put the required source code for this program (RemoteControl.c, uart.c, Config.h, uart.h) in that folder. Then, I modified the makefile in the …/Sourcecode/ folder as shown below:

```
#
#             LUFA Library
#     Copyright (C) Dean Camera, 2014.
#
#  dean [at] fourwalledcubicle [dot] com
#           www.lufa-lib.org
#
# --------------------------------------
#         LUFA Project Makefile.
# --------------------------------------

# MCU Types:
# at90usb1286 for Teensy 2.0++
# atmega16u2 for Arduino UNO R3
# atmega32u4 for Arduino Micro/Teensy 2.0

# TARGET Types:
# ./Bots/<name>/<name>

MCU          = atmega32u4
ARCH         = AVR8
F_CPU        = 16000000
F_USB        = $(F_CPU)
OPTIMIZATION = s
TARGET       = ./Bots/RemoteControl/RemoteControl
SRC          = $(TARGET).c ./Config/Descriptors.c $(LUFA_SRC_USB) ./Bots/RemoteControl/uart.c
LUFA_PATH    = ./LUFA
CC_FLAGS     = -DUSE_LUFA_CONFIG_HEADER -IConfig/
LD_FLAGS     =

# Default target
all:

# Include LUFA build script makefiles
include $(LUFA_PATH)/Build/lufa_core.mk
include $(LUFA_PATH)/Build/lufa_sources.mk
include $(LUFA_PATH)/Build/lufa_build.mk
include $(LUFA_PATH)/Build/lufa_cppcheck.mk
include $(LUFA_PATH)/Build/lufa_doxygen.mk
include $(LUFA_PATH)/Build/lufa_dfu.mk
include $(LUFA_PATH)/Build/lufa_hid.mk
include $(LUFA_PATH)/Build/lufa_avrdude.mk
include $(LUFA_PATH)/Build/lufa_atprogram.mk
```

After the source code is in the correct location and the makefile has been updated to point to it, run “MakeFile.bat” and the program should compile into “RemoteControl.hex”

Loading and running the program

Load “RemoteControl.hex” onto the Teensy using the TeensyLoader. Next, disconnect any other controllers from the Switch and plug in the Teensy. Finally, plug the micro USB into the computer and the CP2104. The Teensy should now accept inputs from the computer.