# Summary

This section is for considerations with regards to the microcontroller being used. It will document the use of a IC Converter vs IC Bridge (USB to SPI) for programming amongst other details for the consideration of the microcontroller. Should give a low level overview of why a certain component / chip is chosen

# Microcontroller Family Considerations

## AVR

Pros:

* Arduino framework can be used with AVR and I’m familiar with it already. Lowers complexity of coding as well as having knowledge on how to program the AVR
* Relatively cheap
* Has a lot of IO pins
* Power consumption is low (according to google)

Cons:

* Performance is middle tier
* Not great for multitasking (not sure if it can handle concurrent video and command relaying)

### ESP

Pros:

* Integrated Wifi and Bluetooth
* Cost is relatively low

Cons:

* Never used ESP before
* Limited IO pins

# AVR

After careful (not really) consideration, the AVR family of microcontrollers is decided on. This is due to familiarity with the ecosystem already, as well as having access to the Arduino ISP for testing.

## Which controller to use

Probably going to be using the ATMega328P but I’ll do some more research into the AVR family depending on our needs

## USB – SPI Interface IC

This is a very important component that is used to program the microcontroller. The plan is to build the ICSP module into the board itself so programming is very easy, but might need to design the ICSP module separately to test.

### Revision 1:

Went full circle essentially. Probably going to borrow a design off the internet to use the programmer. If the IC states that it supports USB 2.0, it works with all future versions as well (USB 3, 4).

There are three protocols to programming a chip UART programming, SPI, and UDPI

UART uses 2, SPI uses 1 ish and UDPI uses 1

UDPI is the newest version of the programming protocols and I have no idea how it works

### Revision 2:

IVE BEEN LOSING MY MIND OVER THIS FOR 3 HOURS AND I THINK I GOT IT

Ecosystem:

Convert data from USB to SPI using an interface chip. Feed this data into a separate MCU, which contains the bootloader for the actual MCU. (Bootloader being the software that runs every time the power gets reset). This bootloader holds the software that facilitates the communication between the two devices using a protocol that is understood by the main MCU and bam coding done. Wtf this took so long :sob: