This LED system accomplishes these tasks, able to receive the letter A-F from the serial monitor and change its brightness with A being 0% and F being 100% brightness, and having a button that can turn the system off or on. When the button is pressed the LED's brightness is set to 0% and ignores all further inputs from the serial monitor. When the button is pressed again it will turn on with the LED's previous light value. The light level is dictated by the MCU on the Arduino which has PWM pins that allow a variable frequency to the LED which can increase or decrease the brightness in contrast to the digital pins which only allow high or low values for the LED. To tell the MCU what to do code is needed which is provided on my Github branch. The button uses a pull-up resistor to create a toggleable ON or OFF state because by default the push button when pressed has a LOW state because of its connection to the ground but when not pressed it also has a LOW state which introduces ambiguity to the MCU to which state the button is in. The pull-up resistor prevents this by allowing a little bit of current to flow to the pin from its 5V source when the button is not pressed for a value of "1" and when the button has pressed a value of "0" therefore reducing ambiguity. Code is also needed to differentiate between those two states and how it interacts with the led, again the code will be on the GitHub branch.

There were multiple problems faced during this project due to my general lack of knowledge of everything needed for this project, but the main two would be the circuitry required and KiCad. The main way I did to get my circuitry correct was firstly knowing that my initial circuit was wrong when I thought it was right. After that, it was a simple search on YouTube on how to program these two things, switchable leds and toggleable buttons. With knowledge of those two things I simply combined them for the circuitry. The main bulk of research (googling) was done for KiCad because learning entirely new software was immensely difficult. The main way I learned was from a playlist created by Digikey hosted by Shawn Hymel, although the tutorial is very dated, almost five years old it laid the groundwork for me on how to understand KiCad. With this basic groundwork knowledge, I delved into specific videos for things that I struggled with, mainly PCB layout. Another minor way I fixed challenges was messing around with KiCad in a separate file to see how things interacted with each other and get a "feel" for the software then implement that back into the Mechatronics Recruit file.

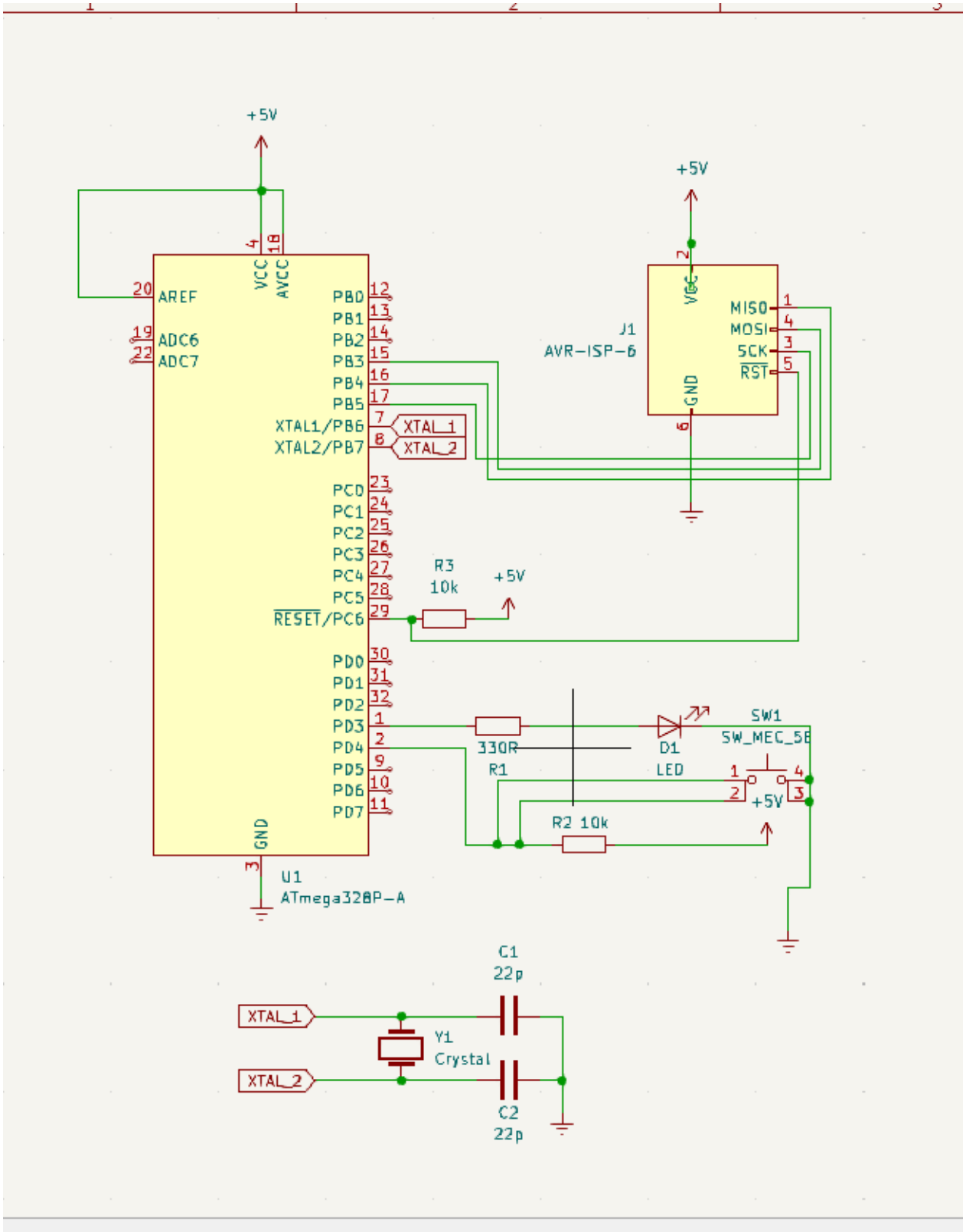
This mechatronics intro project needed a fancy little LED whose light level could be changed via inputs into the serial monitor and turned on and off via a button. Requiring a basic level understanding of circuitry. Transferring this circuitry into PCB required the use of KiCad, a free PCB layout program. This required me to transfer the circuit into schematic form, assign footprints, and then decide the layout of my PCB.

BOM:

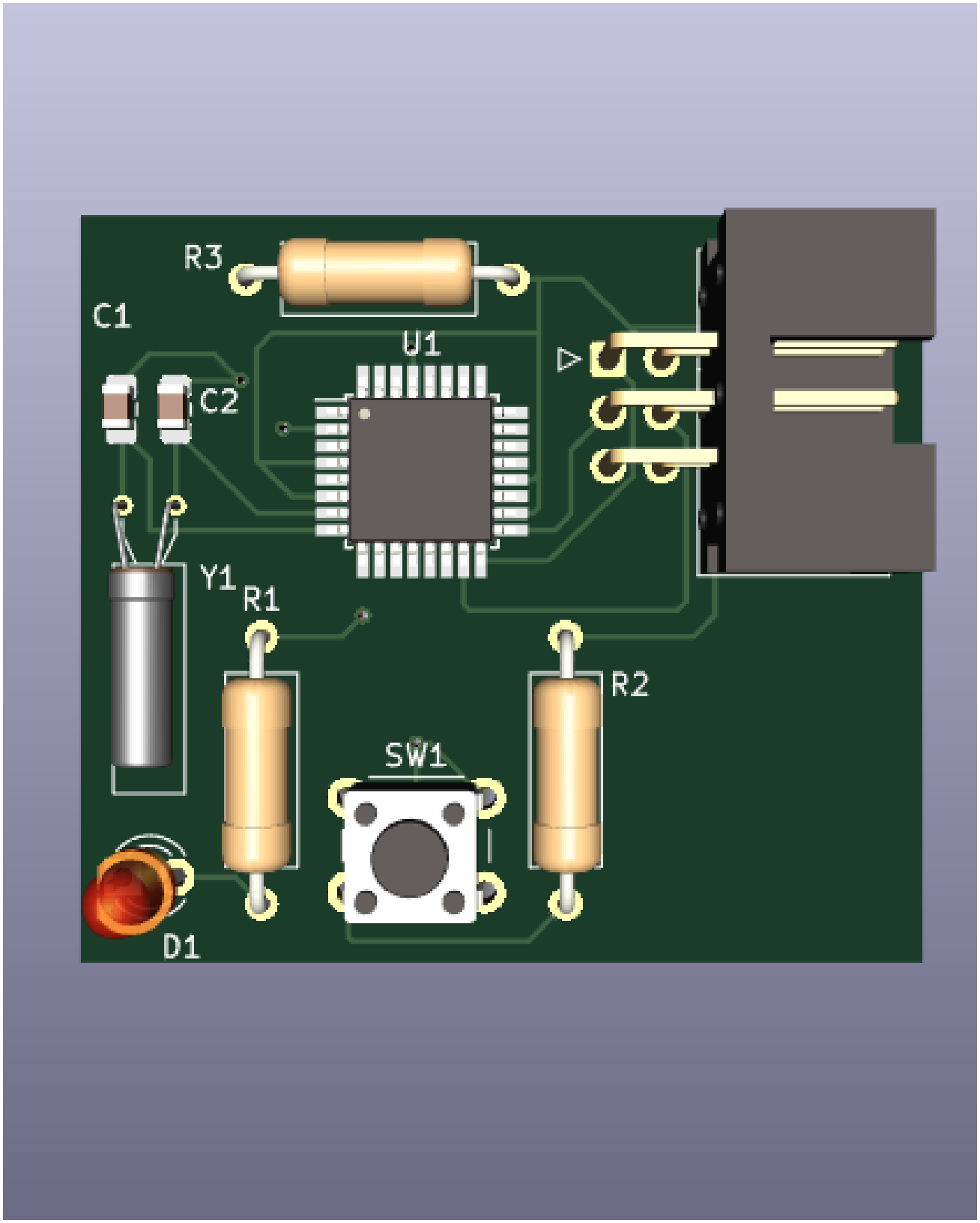
| # | Reference | Quantity | Footprint | |
|---|---|---|---|---|
| 1 | C1, C2 | 2 | Capacitor_SMD:<br>C_0805_2012M | |

| | | | | |
|---|---|---|---|---|
| | | | etric_Pad1.18x1.45mm_HandSolder | |
| 2 | D1 | 1 | LED_THT:LED_D3.0mm | |
| 3 | J1 | 1 | Connector_IDC:IDC-Header_2x03_P2.54mm_Horizontal | |
| 4 | R1 | 1 | Resistor_THT:R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal | |
| 5 | R2, R3 | 2 | Resistor_THT:R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal | |
| 6 | SW1 | 1 | Button_Switch_THT:SW_PUSH_6mm | |
| 7 | U1 | 1 | Package_QFP:TQFP-32_7x7mm_P0.8mm | |
| 8 | Y1 | 1 | Crystal:Crystal_AT310_D3.0mm_L10.0mm_Horizontal | |

Schematic:


+5V
+5V
J1
AVR-ISP-6
VCC
GND
MISO
MOSI
SCK
RST
U1
ATmega328P-A
AREF
ADC6
ADC7
VCC
AVCC
GND
PB0
PB1
PB2
PB3
PB4
PB5
XTAL1/PB6
XTAL2/PB7
PC0
PC1
PC2
PC3
PC4
PC5
RESET/PC6
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
XTAL_1
XTAL_2
R3
10k
330R
R1
D1
LED
SW1
SW_MEC_5E
R2 10k
C1
22p
C2
22p
Y1
Crystal

PCB front:




PCB back: