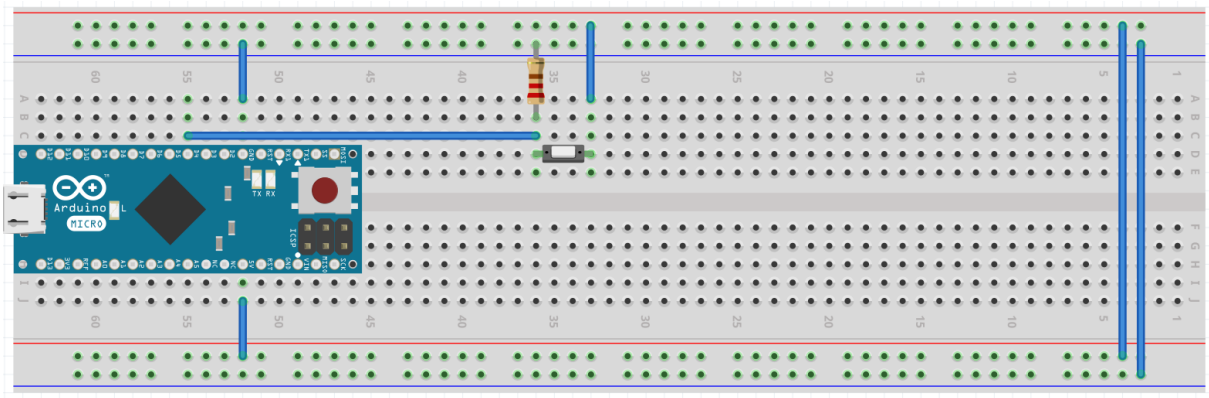
Übungsblatt 4 – Arduino

Für Videos/ Code siehe Ordner „Anhang“ und „Code“

4.1

Button wurde an den im Code vorgegebenen Pin (Pin 4) als Input angeschlossen (Wichtig: Pull-down Widerstand nicht vergessen!). Wenn Button gedrückt wird, wird quasi die Taste „a“ der Tastatur gedrückt.



Anmerkung:

Die Eingabe durch den vorgegebenen Code fühlt sich „unnatürlich“ durch den 1000ms Delay an. Durch folgende Änderung ist die Eingabe „wie gewohnt“.

```
void loop()
{
  Serial.print(digitalRead(BUTTON_PIN));
  // press 'a' as long as the button is down
  if(digitalRead(BUTTON_PIN) == HIGH)
  {
    // we can also send ASCII values
    Keyboard.press('a');
  }
  else{
    Keyboard.release('a');
  }
}
```

4.2

Idee

Für das Spiel Minecraft soll ein eigenes Eingabegerät entwickelt werden, welches die wichtigsten Ingame-Funktionen umfasst. Dazu zählen:

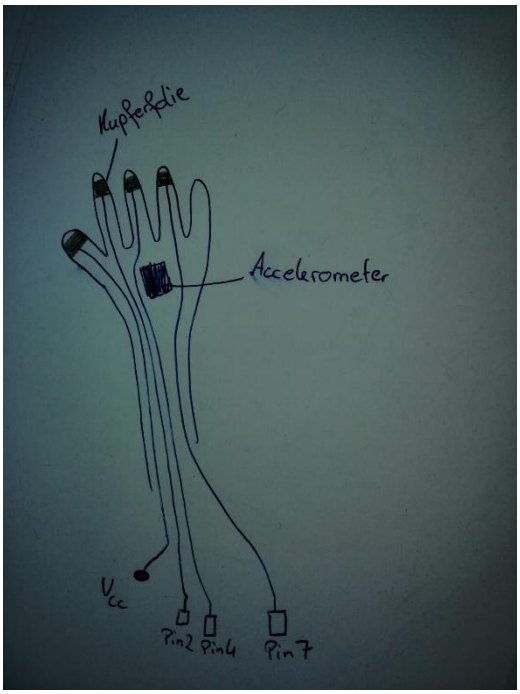
- Laufen (Taste ‚w')
- Umsehen (Maus)
- Abbauen/ Angreifen (linke Maustaste)
- Block setzen (Rechte Maustaste)

Ich habe mich für eine Art „Gaming-Handschuh" entschieden, der die genannten Funktionen abdeckt.

Funktion	**Eingabe normal**	**Neue Eingabe per**	**Am Handschuh**
Laufen	w-Taste	„Button"	Berührung von Zeigefinger und Daumen
Umsehen	Maus	3-Axis Accelerometer	Handschuhh bewegen
Abbauen/ Angreifen	Linke Maustaste	„Button"	Berührung von Mittelfinger und Daumen
Block setzen	Rechte Maustaste	„Button"	Berührung von Ringfinger und Daumen

Umsetzung

1. *Skizzieren/ Visualisieren meines Plans*

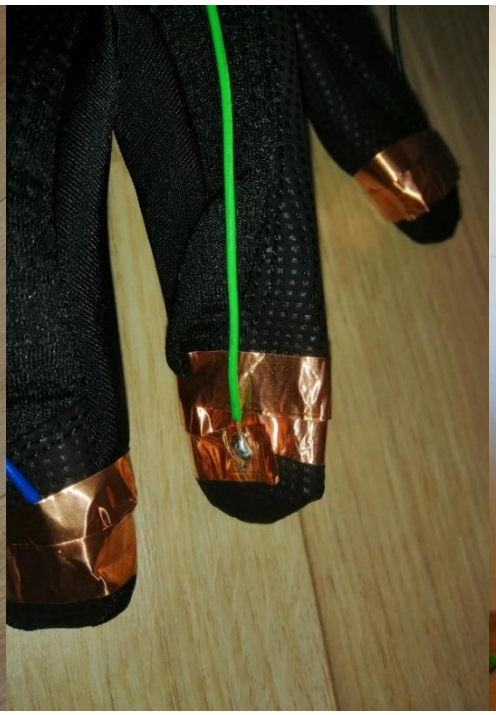


- Der Daumen ist mit der Spannungsquelle verbunden. Sobald einer der Finger mit dem Daumen in Berührung kommt, ist fließt Strom an den jeweiligen Pin des Arduinos. Der Zeigefinger ist soll also an Pin2 anliegen und soll bei Kontakt mit dem Daumen das Event

„Laufen“ bzw. die w-Taste simulieren (gleiches Prinzip für die anderen zwei Finger). Um den Anschluss/ die Funktion des Accelerometers kümmere ich mich später, ich will zunächst erstmals die Buttons zum laufen kriegen.

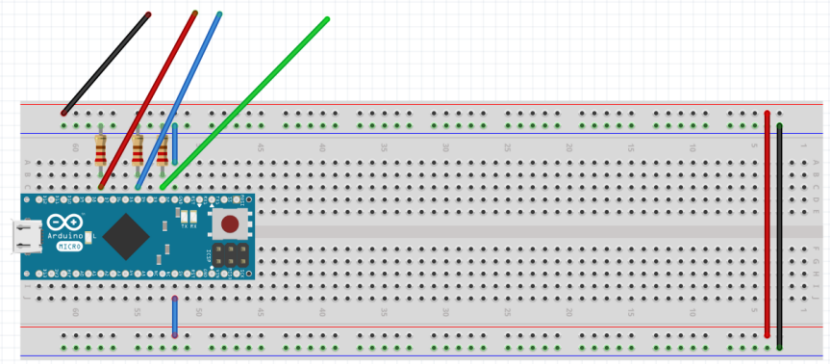
2. *Löten*

- Als nächstes habe ich angefangen, die benötigten Drähte/ Kabel/ Kupferfolien zusammenzulöten, und diese dann auch gleich am Handschuh angebracht (dabei habe ich auch gleich geschaut, dass die verwendeten Litzen lange genug sind, dass man später auch etwas Freiraum in der Armbewegung hat/ falls immernoch zu kurz, kann man einfach noch eine Verlängerung zusammenlöten). Für das Accelerometer habe ich auch gleich drei Stecker verlängert, die ich später für die Daten der x, y, und z-Achse benötige


3. *Schaltung für die Buttons planen und umsetzen*

- Jetzt habe ich versucht, die Schaltung/ Anschlüsse am Steckbrett umzusetzen. Das ging relativ schnell, da die Finger ja jetzt eigentlich einfach nur als Schalter angesehen werden können, die bei Berührung (mit der Spannungsquelle, also dem Daumen) zu einem gewissen Pin Strom leiten sollen, ansonsten nicht. Wichtig war hier aber wieder, die Pull-Down Widerstände nicht zu vergessen, dass kein „schwebender“ Zustand an den Pins vorliegt, wenn kein Strom anliegt, sondern diese klar definiert 0V empfangen.
- Zeigefinger (Grüner Draht) soll an Pin 2, Mittelfinger (Blauer Draht) an Pin 4, Ringfinger (Roter Draht) an Pin 7, Daumen (schwarzer Draht) an Vcc



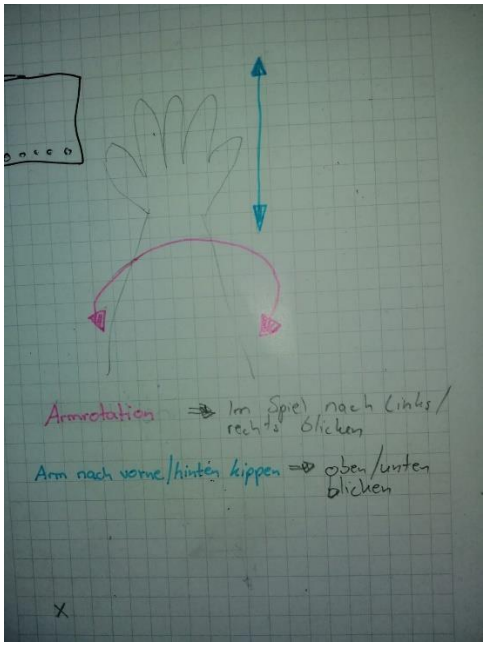
4. *Code für die Fingerbuttons*

- Die Methode checkFingerInput() wird in jedem loop-Durchlauf aufgerufen
- Hier werden die drei Werte für die Finger ausgelesen und dann die passende Aktion ausgeführt
- Ich habe mich für drei aufeinanderfolgende if-else if Schleifen entschieden, weil so mehrere Aktionen gleichzeitig ausgeführt werden können. Ich kann also z.B. in Minecraft gleichzeitig vorwärts laufen und einen Block setzen etc.
- Für die Aktion Block setzen wurde ein Timer eingebaut, sonst ist es relativ unkontrollierbar, wie viele Blöcke man setzt. So dauert es erst immer 200ms bis der Spieler einen neuen Block setzen kann.
- **Anmerkung**: Anfangs konnte man Blöcke nicht abbauen. Die Taste funktionierte zwar, allerdings wurde sie nicht lange genug gedrückt, sondern immer nur ganz kurz und dann sofort wieder released (siehe Video Zwischenupdate.mp4). Das lag daran, dass die click() Funktion der Maus verwendet wurde, die nur einmal Klickt und sofort wieder released. Diese musste lediglich Gegen die press() Methode ausgetauscht werden, bei der man dann selbst bestimmen konnte, wann wieder released werden soll.

```
void checkFingerInput(){
  walk = digitalRead(POINTERFINGER);
  attack = digitalRead(MIDDLEFINGER);
  setBlock = digitalRead(RINGFINGER);
  if(walk == HIGH && ! isWalking){
    isWalking = true;
    Keyboard.press(KEY_WALK);
  }
  else if(walk == LOW){
    isWalking = false;
    Keyboard.release(KEY_WALK);
  }
  if(attack == HIGH && ! isAttacking) {
    isAttacking = true;
    Mouse.press();
  }
  else if(attack == LOW){
    isAttacking = false;
    Mouse.release();
  }
  if(millis() - lastMillis2 >= SETBLOCK_DELAY){
    lastMillis2 = millis();
    if(setBlock == HIGH) {
      Mouse.click(MOUSE_RIGHT);
    }
  }
}
```

5. *Accelerometer verstehen und testen*

- Für mein Eingabegerät will ich, dass…
  … der Spieler nach links/ rechts blickt, wenn der Arm nach rechts/ links rotiert wird
  … der Spieler nach oben/ unten blick, wenn der Arm nach vorne/ hinten gekippt wird
  ➔ Dafür will ich das Accelerometer hernehmen



Notizen zur Funktion des Accelerometers:

- damit kann man die Beschleunigung auf die 3 Raumachsen (x,y,z) messen → Ausrichtung steht auf dem Acc mit drauf.
- in unserem Fall 3-5V Inputspannung akzeptabel
- 3 Pins für jede Achse als Output
- Wenn z-Achse nach oben schaut, wirkt die Gravitation auch nur auf die Z-Achse
  ➔ Resultat der eingehenden Spannung: x: 344, y: 338, z: 416
  ➔ 416 entspricht ca. 2V
  ➔ 340 entspricht ca. 1.65V
- Wenn y-Achse nach oben schaut: x: 348, y: 405, z: 346
- Wenn x-Achse nach oben schaut: x: 409, y: 335, z: 346
  ➔ Wenn x, y- oder z-Achse nach unten schauen, sind Werte von ca. 265 (1.3V) zu erwarten
  ➔ Das sind allerdings nur die Werte, die wir bekommen, wenn sich der Sensor in Ruhe befindet. Bei Bewegung wirkt dementsprechend mehr oder weniger Kraft auf die Achsen
  ➔ Im freien Fall wird für einen kurzen Moment keine Beschleunigung an den 3 Achsen gemessen/ bzw. Werte von ca. 335
- Wenn ich Sensor entlang der x-Achse um 45 Grad kippe, erhalte ich für y- und z- nahezu identische Werte, da die Kraft auf beide Achsen gleich ist

Anschließen des Accelerometers an den Arduino/ ans Breadboard:

- Relativ intuitiv, Eingang für Vcc, Ausgang für Ground, sowie drei Ausgänge für x, y und z-Wert vorhanden. Die Werte müssen einfach nur an passende analoge Pins des Arduinos angeschlossen werden (ich wähle A1, A2, A3)

Lösung:

- Ich baue das Accelerometer so auf den Handschuh, dass im Normalfall (ausgestreckter Arm, keine Rotation oder Neigung) die z-Achse nach oben schaut
- Die x-Achse hätte dann einen Werte von ungefähr 340, die y-Achse genauso
- Wenn ich jetzt meine Hand nach links/rechts rotiere, ändert sich theoretisch nur der Wert der x-Achse.
- Wenn ich meine Hand nach vorne oder hinten neige, dann ändert sich nur der y-Achsen-Wert
- Sobald ein Threshold-Wert überschritten wird/ ich also genug Änderung an der x- oder y-Achse habe, bewege ich die Maus in die entsprechende Richtung (z-Achse kann komplett ignoriert werden)
- Damit ich die richtige Mausgeschwindigkeit finden konnte, die sich einigermaßen benutzen lässt, habe ich einfach herumprobiert, bis ich einen meiner Meinung nach passenden Wert hatte, der nicht zu einer kompletten Übersteuerung führt, aber auch nicht ewig langsam ist.

```
void moveMouse(int x, int y){
  if(x > UPPER_THRES){
    Mouse.move(-MOUSE_SPEED, 0, 0); // move mouse right
  }
  else if(x < LOWER_THRES){
    Mouse.move(MOUSE_SPEED, 0, 0); // move mouse left
  }
  if(y > UPPER_THRES){
    Mouse.move(0, MOUSE_SPEED, 0); // move mouse down
  }
  else if(y < LOWER_THRES){
    Mouse.move(0, -MOUSE_SPEED, 0); // move mouse up
  }
}
```

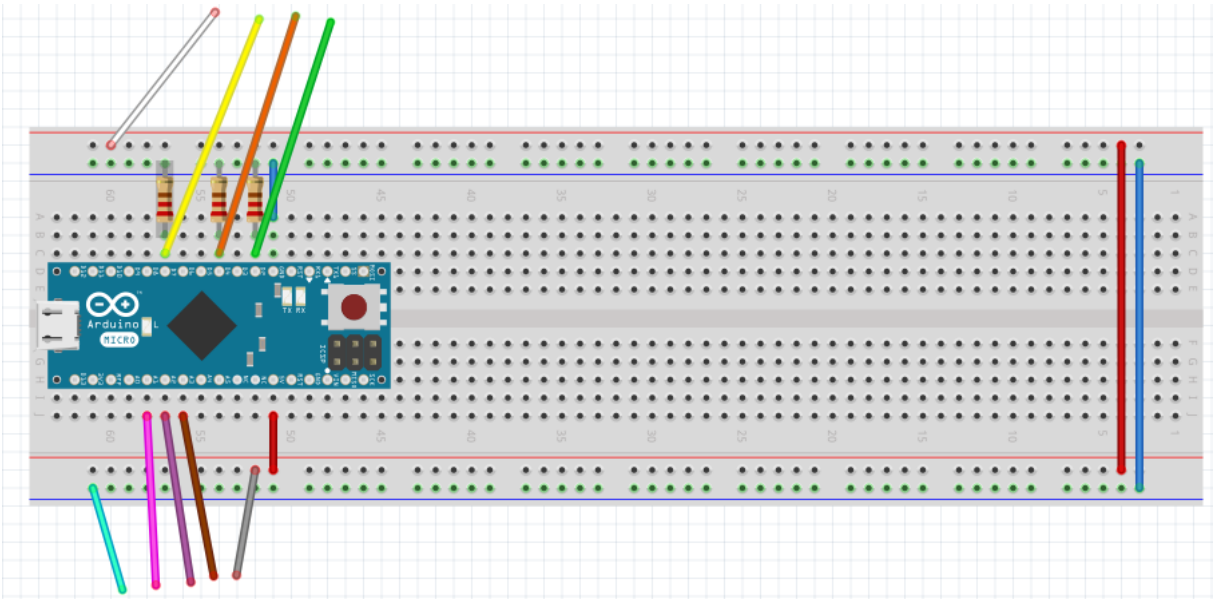
Notizen:

- Zunächst hatte ich versucht, in jedem loop() Durchlauf einen Wert für x und y auszulesen und diesen dann vom vorherigen x und y Wert zu subtrahieren, um die Differenz zum vorherigen Zustand zu berechnen und dann den Mauscursor basierend auf diesen Werten zu bewegen. Allerdings gab es hier mehrere Probleme. Zum einen hatte der Curser selbst im Ruhestand des Sensor „herumgewackelt“, da immer wieder kleinste Änderungen an dem Accelerometer wahrgenommen wurden (siehe Video „Sensor Update 1.mp4“). Das Problem konnte sich einigermaßen gut beheben lassen, indem ich eine Art Deadzone in den Code eingebaut habe, damit der Sensor erst um einen gewissen Grad bewegt werden musste, damit die Maus reagiert. Das hat auch geklappt (siehe Video „Sensor Update 2.mp4“), allerdings war die ganze Steuerung mit dieser Methodik allgemein sehr unzuverlässig und schwer bedienbar. Deshalb habe ich mich dann dazu entschieden. Die Maus immer dann in eine Richtung zu bewegen, sobald ein bestimmter Threshold-Wert für x und y überschritten wird.

6. *Zusammenfügen aller Elemente*

- Der finale Code, sowie ein Video zum Resultat sind im Anhang zu finden
- Zunächst wurde das Accelerometer am Handschuh angebracht (mit Klebeband), möglich so, dass die z-Achse bei ausgestrecktem Arm senkreckt zum Boden/ nach oben zeigt

- Abkleben offener Lötstellen der zusammengelöteten Litzen (hatte zu dem Zeitpunkt noch keine Schrumpfschläuche)
- Anschließend wurde alles ans Breadboard angesteckt



➔ Weiß: Daumen
➔ Gelb: Ringfinger
➔ Orange: Mittelfinger
➔ Grün: Zeigefinger
➔ Cyan: Ground für Accelerometer
➔ Rosa: x-Wert des Accelerometers
➔ Lila: y-Wert des Accelerometers
➔ Braun: z-Wert des Accelerometers (wird allerdings nicht benötigt)
➔ Grau: Spannungsversorgung für Accelerometer