### autonom

Um den Parkour autonom zu absolvieren wurden zusätzliche Sensoren verwendet, mit denen diese Aufgabe vereinfacht wird. Zum Einen wurde ein Gyroskop angeschlossen um die Orientierung im Raum zu vereinfachen. Zum Anderen ein Ultraschallsensor damit Hindernisse erkannt werden können.

Für die Programmierung wurde der Raspberry Pi und der Arduino verwendet. Auf dem Arduino ist die Motorsteuerung gespeichert und auf dem Pi die Kamerafunktion und der Array. Beide Computer kommunizieren mithilfe einer i2c-Verbindung und senden Zahlen hin und her, welche die verschiedene Funktionen auslösen. Das autonome Fahren beim Auto konnte leider nicht komplett absolviert werden, da das Auto selber nicht in einer Linie fährt, sondern eine Rechtsneigung besaß. Es fuhr so lange vorwärts, bis sich ein Hindernis 30 Zentimeter vor dem Fahrzeug befindet. Dieser Wert wird mit einem Ultraschallsensor berechnet. Dieser ist am Arduino angeschlossen und muss dem Raspberry Pi mitteilen, dass ein Hindernis vorhanden ist und ein Foto gemacht werden muss. Durch die i2c-Verbindung wird vom Arduino die Zahl 7 versendet. Das Problem hierbei ist, dass der Arduino nicht von sich aus Zahlen senden kann. Somit muss der Pi regelmäßig den Arduino fragen, ob ein Hindernis vorhanden ist. Diese Frage wird mithilfe einer Zahl jede Sekunde gestellt und geantwortet wird mit einen von zwei Zahlen. Entweder ist ein Hindernis vorhanden oder die Bahn ist frei. Nachdem der Pi die Antwort erhalten hat wird dementsprechend gehandelt. Bei einem Objekt wird die Kamerafunktion ausgelöst und das Bild wird auf Twitter geladen. Nachdem das Foto geschossen wurde, wird eine Rechtskurve gefahren und dann fährt das Auto weiter.

Die Motorsteuerung wurde aus dem Ersten Projekt im 1. Semester übernommen und angepasst. Um Platz auf dem Breadboard und Pins auf dem Arduino zu sparen wurde nur eine H-Brücke gebaut und die linken bzw. rechten Reifen parallelgeschaltet. Da die Motoren sehr schwach sind, wurde immer der höchste Wert von 255 verwendet. Bei den Links- und Rechtsdrehungen wird wie ein Panzer gefahren. Das heißt, dass beide Seiten gegenfahren bis sich das Auto in einem Winkel von 90 Grad gedreht hat.

### ferngesteuert

Bei der Aufgabe das Fahrzeug über den Parkour per Fernsteuerung zu fahren gibt es viele Möglichkeiten die Fernsteuerung zu kreieren. Wir haben uns dann schlussendlich für die Laptop Tastatur entschieden. Dies hat den Vorteil, dass jeder den Code testen kann ohne eine physische Fernsteuerung mitnehmen zu müssen und falls es Probleme bei der Verbindung gibt kann die Mohne per Not-Stopp ausgeschaltet werden. Da das Fahrzeug bereits mit einem Samba Server verbunden ist war die Remote Steuerung leicht zu verwenden und in der Testphase konnten die verschiedenen Programme schnell verändert werden. Der Code für die Fernsteuerung heißt ‚FernFoto.py‘ und kommt von ‚Fernsteuerung‘ und ‚Foto‘.

Die Motorsteuerung war auf dem Arduino gespeichert. Dies war bei der ursprünglichen Mohne, wie auch bei unserem endgültigen Auto der Fall. Auf die Motorsteuerung selber wurde bereits im Abschnitt Autonom eingegangen.

Der Code für die Fernbedienung wurde auf dem Raspberry Pi mithilfe eines Keyloggers geschrieben. Der verwendete Logger kommt aus die pynput Library. Dies hatte den Vorteil, dass es auch eine Key Release Funktion gab und dadurch die Mohne stoppt, wenn eine Taste losgelassen wurde. Der Nachteil war allerdings, dass der Keyloggers nicht zusammen mit Schleifen funktioniert. Somit konnte der Raspberry Pi nicht selbständig Fotos schießen. Dieses Problem wurde gelöst indem ein separater Tastendruck die Kamera auslöst und das Bild auf Twitter hochlädt.

Die Mohne wird klassisch mit den Pfeiltasten gesteuert. Beim Drücken der Leertaste entsteht das Foto und mit der Escape Taste wird das Programm gestoppt. Wenn keine Taste gedrückt wird oder eine losgelassen wird werden die Motoren ausgeschaltet.

Jetzt müssen die Befehle nur noch an den Arduino geschickt wird, damit diese umgesetzt werden. Die im Unterricht gelernte i2c-Verbindung wurde verwendet und modifiziert. Jede Taste wurde einer Zahl zugeordnet, die an den Arduino geschickt wird. Wenn die Motoren stoppen sollen wird auch eine Zahl gesendet.

Der Code auf dem Arduino heißt ‚Fern\_1.0‘. Auf diesem wurden bereits Klassen für die jeweiligen Richtungen geschrieben. Also vorwärts, rückwärts, rechts und links. Dazu natürlich noch die Stopp Funktion. Durch die i2c-Verbindung wartet der Arduino auf die Zahlen, welche vom Raspberry Pi gesendet werden. Bei der Zahl 1 wird nach vorne gefahren. Bei der 2 nach rechts. Wird die 3 gesendet nach Links und bei der 4 rückwärts. Die Nummer 5 wird für die Stopp Funktion verwendet.

Die Fernsteuerung lief trotz vielen Schnittstellen sehr präzise und ohne Verzögerung.