

Wintersemester 2018/19

Fachbereich 4

Angewandte Informatik

MM Entwicklung von Multimediasystemen

Prof. Dr.-Ing. Johann Habakuk Israel

Sebastian Keppler

Interaktives Modellhaus

Vorgelegt von:

Gruppe Y

|  |  |
| --- | --- |
| Rico Eisenberg |  |
| Moritz Fabian Engers | s0559037 |
| Johann Jakob Gillhoff | S0559832 |
| Tony Nguyen Tien |  |

Berlin, den

Inhalt

[Interaktives Modellhaus – Idee 2](#_Toc534990348)

[Systembuild 2](#_Toc534990349)

[Handgesten 2](#_Toc534990350)

[Kamera Positionen ändern 2](#_Toc534990351)

[Regen aktivieren 2](#_Toc534990352)

[Serielle Schnittstelle 3](#_Toc534990353)

[Sendende und Empfangende Struktur vom Arduino 3](#_Toc534990354)

[Sendet Arduino zu Unity 3](#_Toc534990355)

[Empfängt Arduino von Unity 3](#_Toc534990356)

[Umsetzung 3](#_Toc534990357)

[Erstellung des Realen Modellhauses 3](#_Toc534990358)

[Aufbau des Hauses 3](#_Toc534990359)

[Programmierung des Arduinos 4](#_Toc534990360)

[Erstellung des virtuellen Modellhauses in Unity 4](#_Toc534990361)

[Erstellung eines 3D Modells 4](#_Toc534990362)

[Auswirkungen auf das 3D Modell erstellen 4](#_Toc534990363)

[Bewegung im/am Haus 4](#_Toc534990364)

[Veränderndes Wetter 4](#_Toc534990365)

[Kamera Wechsel 4](#_Toc534990366)

[Erfassung der Leap Motion Gesten 4](#_Toc534990367)

[Controller: Der alles Verbindet 5](#_Toc534990368)

[Aufgetretene Probleme 5](#_Toc534990369)

[Gyroskop 5](#_Toc534990370)

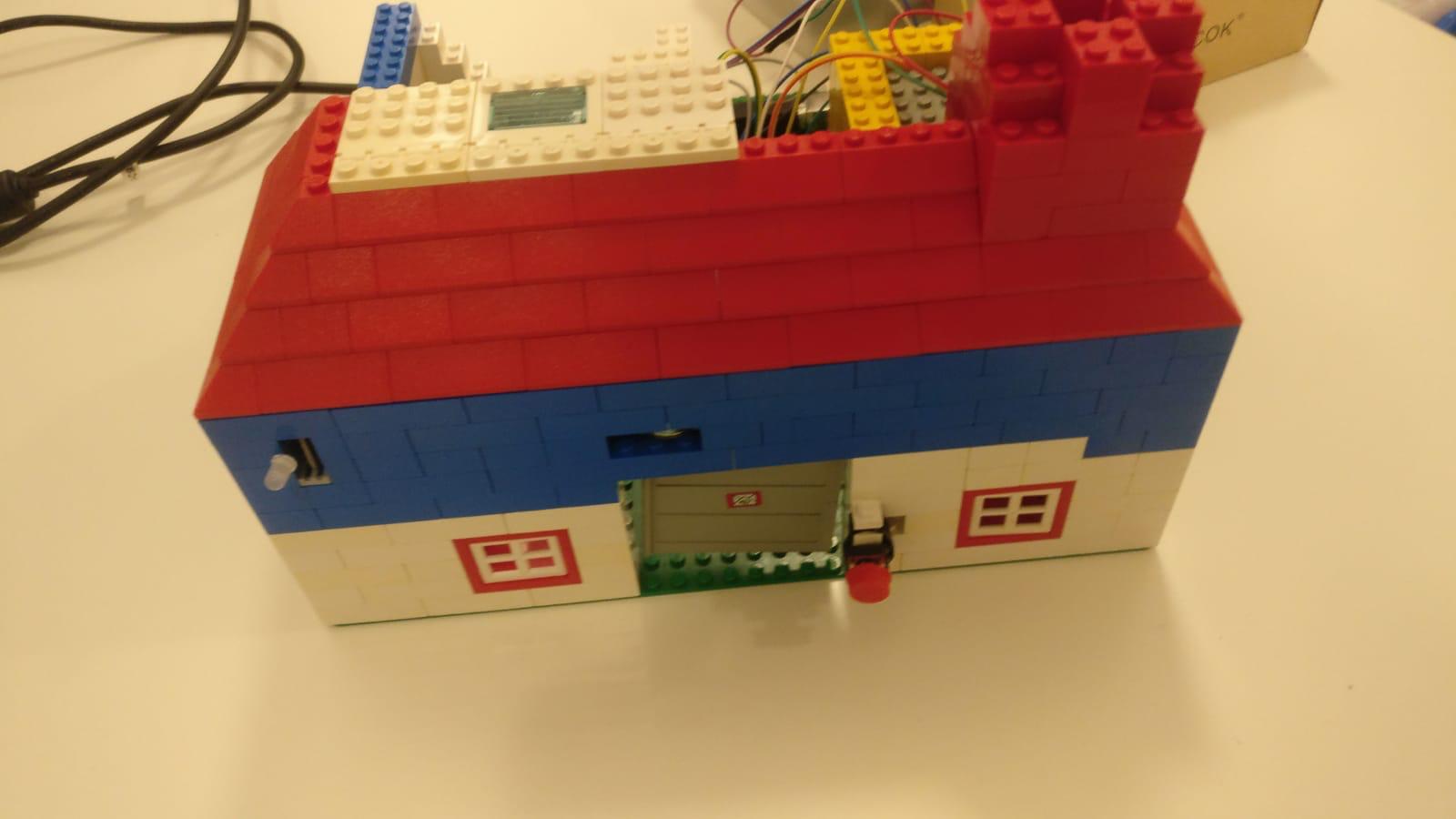
[Git 5](#_Toc534990371)

[Problemeeee 5](#_Toc534990372)

[Quellen 6](#_Toc534990373)

# Interaktives Modellhaus – Idee

# Systembuild



Handgesten

Serielle Schnittstelle

## Handgesten

Es gibt zwei Möglichkeiten mit der Leap Motion zu interagieren:

### Kamera Positionen ändern

Man kann die Positionen der Kamera ändern, indem man mit seien Fingern die Gesten für eins, zwei und drei macht und so zwischen den Kameras hin und her wechselt.

### Regen aktivieren

Hierzu muss man eine Regengeste machen. Dazu zeigen die Finger nach unten und bewegen sich wiederholend leicht nach vorne und hinten.

## Serielle Schnittstelle

* Tür → „potiTuer“
* Klingel → „klingelState“
* Kellertür Sonar → „sonarCM“
  + Kellertür Alarm → „isAlarm“
  + Blinklicht activ with isAlarm
* Magnet → „magnetState“
* Fenster → „lichtsensorState“
* Gyroskop → „x, y, z“
* Servo ← incomingByte

### Sendende und Empfangende Struktur vom Arduino

#### Sendet Arduino zu Unity

„potiTuer, klingelState, sonarCM, isAlarm, magnetState, lichtsensorState, x, y, z“

Die Werte werden als ganzer String übersendet an Unity.

#### Empfängt Arduino von Unity

incommingByte

Das Byte wird dabei als ASCII Zeichen interpretiert und daher ist eine übersendete „0“ eine 48 und wird so interpretiert. Daher muss von Unity einen „1“ oder „0“ übersendetet werden.

# Umsetzung

Um unser Projekt zu verwirklichen haben wir es untereilt umso produktiv in der Gruppe daran Arbeiten zu können. Dazu haben wir das Projekt folgender Maßen geteilt:

## Erstellung des Realen Modellhauses

### Aufbau des Hauses

Nachdem wir uns klar gemacht haben, welche Funktionen das Haus haben soll, planten wir wie wir das Umsetzten können und welche Sensoren und Motoren wir dafür bräuchten. Außerdem mussten wir uns noch entscheiden, aus welchem Material wir unser Haus fertigen wollen. Zur Diskussion standen Holz, Pape und Lego. Wir haben uns dann für Lego entschieden, da wir das Material zum einen Vorrätig hatten, außerdem zeichnet es sich durch seine leichte Verarbeitbarkeit und Vielseitigkeit aus.

Nachdem alles Benötigte organisiert war, begannen wir das Haus aus Lego zu bauen und dabei versuchten wir bestmöglich die Sensoren, den Servo und die Leap einzubauen. Nachdem wir dann auch noch alle Sensoren und den Servo and den Arduino angeschlossen hatten und die Verkabelung dokumentiert hatten, begannen wir den Arduino zu programmieren.

### Programmierung des Arduinos

Diese verlief relativ reibungslos und schneller als erwartet. Dies lag unter anderem daran, dass in dem von uns bestelltem Sensor-Kit bereits Beispielcode zu jedem Bauteil beilag. Nach einiger Anpassung lief das meiste recht schnell und wir konnten wie in dem SL gelernt über die serielle Schnittstelle die Sensordaten über die Serielle Konsole senden. Unity empfängt und verarbeitet diese dann um die Änderungen am Modellhaus in Unity darzustellen.

### Empfang und Verarbeitung der Daten in Unity

Da der serielle Empfang einen blockierenden Aufruf enthält, haben wir uns dazu entschlossen die serielle Kommunikation in einem eigenen Thread durchzuführen. So können wir sicherstellen, dass eine hohe FPS gewährleistet bleibt. Außerdem wird der Haupt Unity Thread nicht blockiert, wenn die Serielle Verbindung abbricht. Die gesamte serielle Kommunikation lagern wir in ein Script aus welches wir dem GameObject „SerialCommunicator“ hinzufügen. Dieses GameObject existiert nur einmal und kann von anderen Komponenten für die Kommunikation genutzt werden. Innerhalb des Scripts benutzen wir das C# volatile Keyword. Da wir auf die Attribute der Klasse von mehreren Threads aus zugreifen. Jede Zeile, welche vom Arduino über die Serielle Schnittstelle übertragen wird, liefert den gesamten State des Modellhauses. Diese Komma-Separierte Zeile wird aufgeteilt und die jeweiligen Werte werden aktualisiert.

## Erstellung des virtuellen Modellhauses in Unity

### Erstellung eines 3D Modells

Nachdem das Haus aufgebaut war, konnten wir eine 3D Modell in [Programm einfügen]

### Auswirkungen auf das 3D Modell erstellen

Nachdem das 3D Haus in Unity importiert war mussten wir noch die Umgebung auf den Userinput reagieren lassen. Dazu haben wir zunächst die Auswirkungen erschaffen, die der User auf das 3D Modell haben kann.

#### Bewegung im/am Haus

#### Veränderndes Wetter

#### Kamera Wechsel

### Erfassung der Leap Motion Gesten

Nun mussten wir uns noch darum kümmern, dass die Hände über dem reellen Hause erfasst werden.

### Controller: Der alles Verbindet

Damit bei einer Eingabe auch was passiert, müssen wir nun noch die passende Eingabe mit der dazu gehörigen Auswirkung Logisch in Unity verbinden. Dazu…

# Aufgetretene Probleme

## Gyroskop

Zunächst ließ sich das Gyroskop nicht auffinden in dem von uns bestellten Sensorkit, da dieser ungünstig Positioniert war, sodass wir ihn erst eine Woche später als eigentliches Gyroskop identifizieren konnten und einbauen konnte. Da das Bauteil schließlich doch noch gefunden wurde, war es ein eher kleineres Problem.

Das größere war die Programmierung des Arduinos zum auslesen der Daten vom Gyroskop.

## Git

Wir hatten uns für GitHub entschieden um unseren Fortschritt uns gegenseitig zu teilen. Da wir alle noch recht unbewandert waren mit Git, hat uns das uns ein paar Mal ein paar Steine in den Weg geworfen. Wir konnten uns jedoch immer noch weiter helfen.

## Problemeeee

# Quellen

|  |  |  |
| --- | --- | --- |
| Bezeichnung | Link | Für was benutzt? |
| Gyro | https://playground.arduino.cc/Main/MPU-6050 | Arduino Programmierung |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |