Bewegungsmelder mit Log

Pflichtenhelft & Dokumentation

Teko 2024 Mikroprozessortechnik

Simren Singh

Inhalt

Version	Datum	Bearbeiter	Bemerkungen
1.0	11.03.2024	Simren Singh	Finished Version

1 Pflichtenhelft

Für den Mikrocomputertechnik-Unterricht bei der Teko im Jahrgang 2024 welche vom Lehrer Chrisitian Meier Unterrichtet wird, müssen wir ein kleines Projekt kreieren.

1.1 Ausgangslage

Bei meinem Haus haben wir einen Kellereingang ohne Schloss. Jeder könnte da etwas verstecken oder klauen, ohne das wir etwas davon mitbekommen.

1.2 Initialkritierien

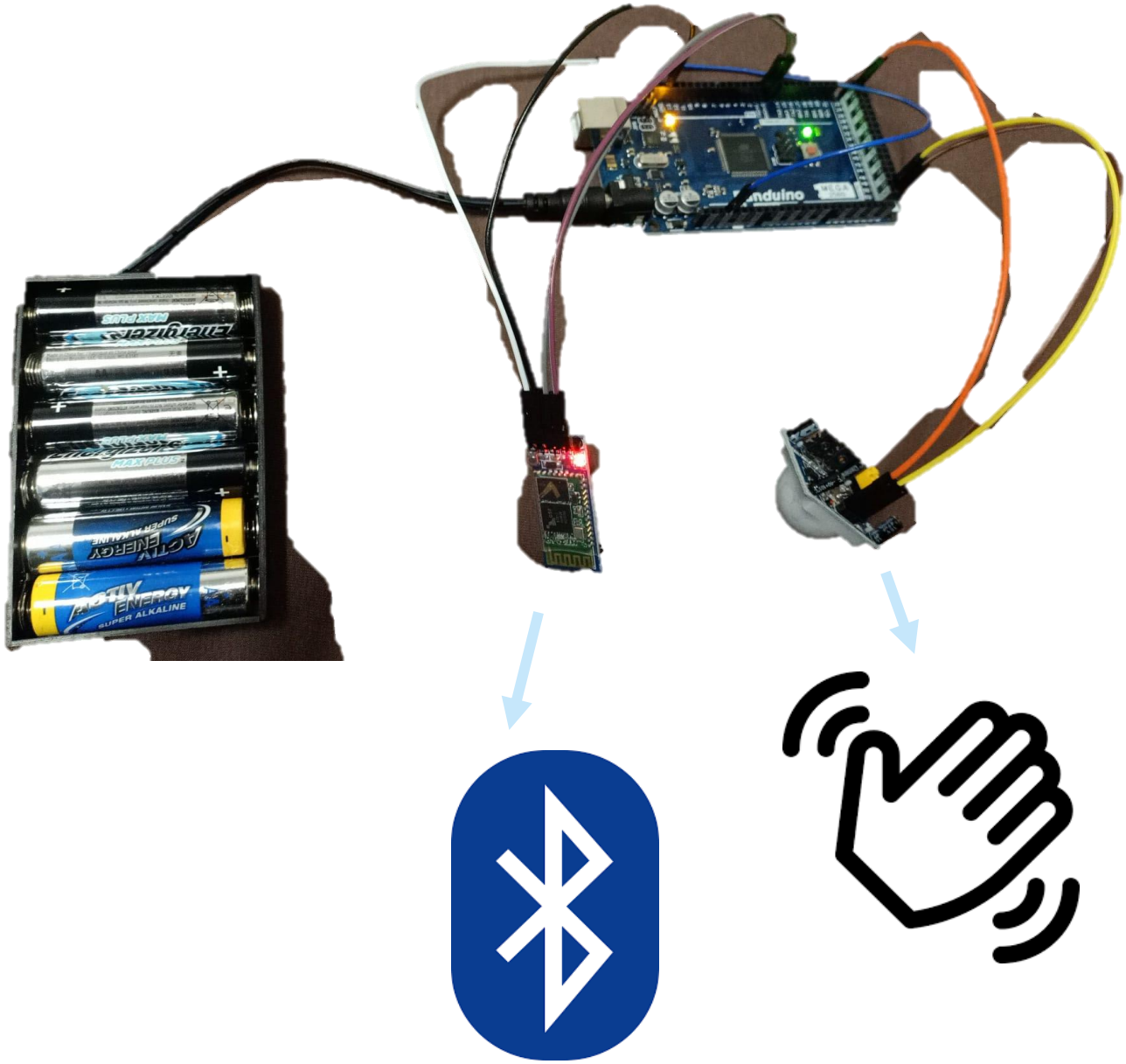
Aktivitäten sollten mit DateTime-stempel geloggt werden. Anhand dieser kann geprüft werden, dass es keine unerwarteten Aktivitäten hab, und falls schon, herausfinden um welche Zeit und wie regelmässig.

Diese Aktivität-Protokoll sollten anhand Bluetoothgerät (Handy) aufrufbar sein.

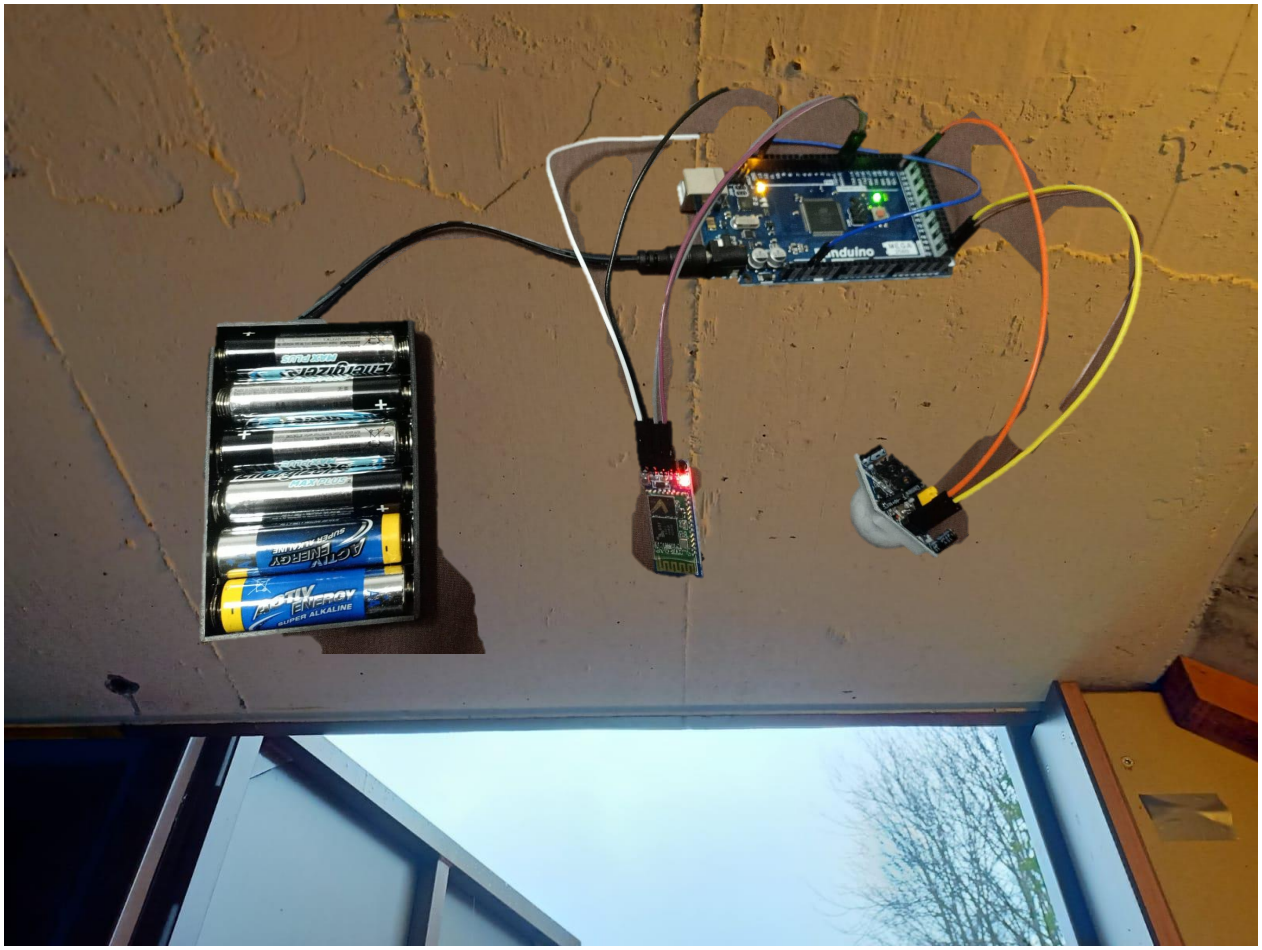
Alle Komponenten sollten im Funduino-Baukasten verfügbar sein und nicht andere Komponente nötig haben.

1.3 Lösung Bewegungsmelder

Ein Bewegungsmelder sollte hier Zeitstempel zu Aktivitäten speichern. Diese sind per Bluetooth aufrufbar.



Dieses Sollte dann an der Decke der Eingangs befestigt werden.



1.4 Hardwarebestandteile

1x Funduino MEGA 2560

1x Bluetooth HC-05

1x Bewegungsmelder HC-SR501

1x Batteriestecker

6x AA Baterien

8x div Kabel

1x Funduino-To-USB (Für initial upload)

1.5 Kriterium-Soll-Ist

Kriterium	Soll	Ist
Pflichtenheft erstellen	Pflichtenheft mit Ausgangslange/Kriterien/Abgrenzung	Alle Titel im Dokument
Terminplan	Terminplan vom Projekt	Simple & nur theoretisch. Wie es sein sollte, nicht wie es wirklich war
Elektronik-Schema	Fritzing elektronik schema	Fritzing elektronik schema
Flussdiagramm von SW-Modulen	Flussdiagram mit sichtbaren Abgrenzungen pro SW-Modul	Flussdiagram mit sichtbaren Abgrenzungen pro SW-Modul
SW-Regeln	camelCase	camelCase ausser für Struct & Grosse CONST
Clean-Code	Clean Code	Bin mir nicht sicher obs clean ist…
3rd-party-code korrekt Anerkennen	extern kopierten Code mit Author-comment anerkennen	getMemoryFunkition an User ec2021 anerkannt
Modularisieren	Code ist Modularisiert	src.ino - Mainfile mit MovementSensor MemoryManagment.h - MemoryFunktion Logger.h - Logging Logik SerialCommands.h - Momentan nur Serial.Read(, aber zukünftig erweiterbar mit z.B. debugmessage anhand global var isDebug & Funktion für automatischen PrintLn nach Print.)
Testprotokoll	Testprotokol erstellen	Last-Minute und mageres Testprotokoll
Versionierung	Versionierungstool verwenden	Github:https://github.com/eternalCODER420/tekoArduino
Besonderes Feature	Loggingdatum Liste	Anhand pointer linkedlist erstellt anhand einmaligen setBaseTime() anfangs Programm und millis() timestamp erstellen können
In grenzen des Fundino Baukasten bleiben	Projekt mit Fundoino Bauteilen realisieren	Projekt mit Fundoino Bauteilen realisiert
Bewegungungen in Log speichern	~~Log in grossen Array speichern~~ Bewegung werden als Liste im System gespeichert	Mit LinkedList gelöst

Zeit Loggen	Im Bewegungslog sollte immer Zeit stehen	Funtioniert nur für Jahr 2024, da Tage pro Monat hartcodiert. In Zukunft alles mit "Real-Time Clock (RTC)"-Gerät ersetzen.
Overflow Log vermeiden	~~Array bei letzter position loopen~~ Funktion finden um Speicher zu messen Letzte Einträge bei Liste löschen bis genug speicher	Funktion zum Speicher messen vorhanden. Aber: Logging wird gestoppt bis User Liste cleart. User kann solange einfach jede Woche Liste manuel leeren, bis dies entwickelt wurde.
Coding richtlinien für sparen von Speicher verfolgen	Folgenden Rat befolgen: https://support.arduino.cc/hc/en-us/articles/360013825179-Reduce-the-size-and-memory-usage-of-your-sketch	Möglichst kleine Datentypen Möglichst lokale Variabeln
Overflow UserInput vermeiden	Sicherheitslogik um Input ab bestimmter grösse zu ignorieren	UserInput auf 4 begrenzt. Alles danach wird einfach am User über Console zurückgeschrieben und nicht weiter verarbeitet.

1.6 Abgrenzung

Wegen der Deadline müssen Grenzen gesetzt werden, damit dies konkurrent zu allen anderen Projekten Zeitmässig abgeschlossen werden kann.

1. Logging sollte nur für Jahr 2024 funktioneren, damit die Komplexität bis zur Deadline umsetzbar ist.
2. Es sollte alles mit Fundino-bauteilen möglich sein.
3. Die Schönheit der Consolenausgaben wird ignoriert.
4. Es gibt keinen IsDebug code (wie debug-messages).

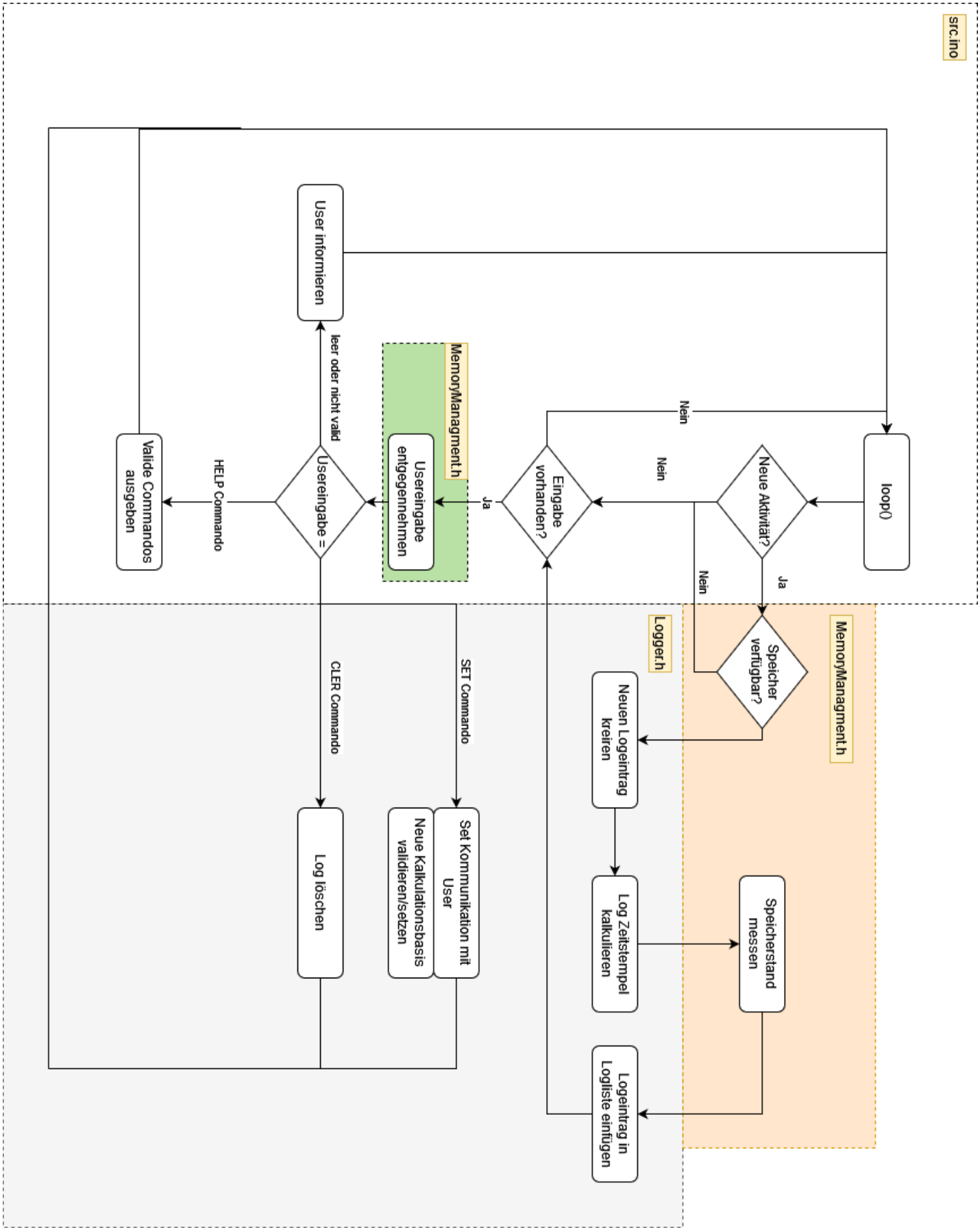
1.7 Terminplan

Start	Ende	Task
19.01.2024	24.02.2024	Brianstorming Projektidee
28.01.2024	04.02.2024	Prototyp Bluetooth
03.02.2024	11.02.2024	Prototyp MovementSensor
11.02.2024	18.02.2024	Log Recherche
17.02.2024	18.02.2024	Linked List
19.02.2024	24.02.2024	Timestamp Logik
24.02.2024	25.02.2024	Memory Recherche
24.02.2024	25.02.2024	Memory Logik
26.03.2024	11.03.2024	Dokumentation
02.03.2024	09.03.2024	Diagramme
09.03.2024	11.03.2024	Präsentation

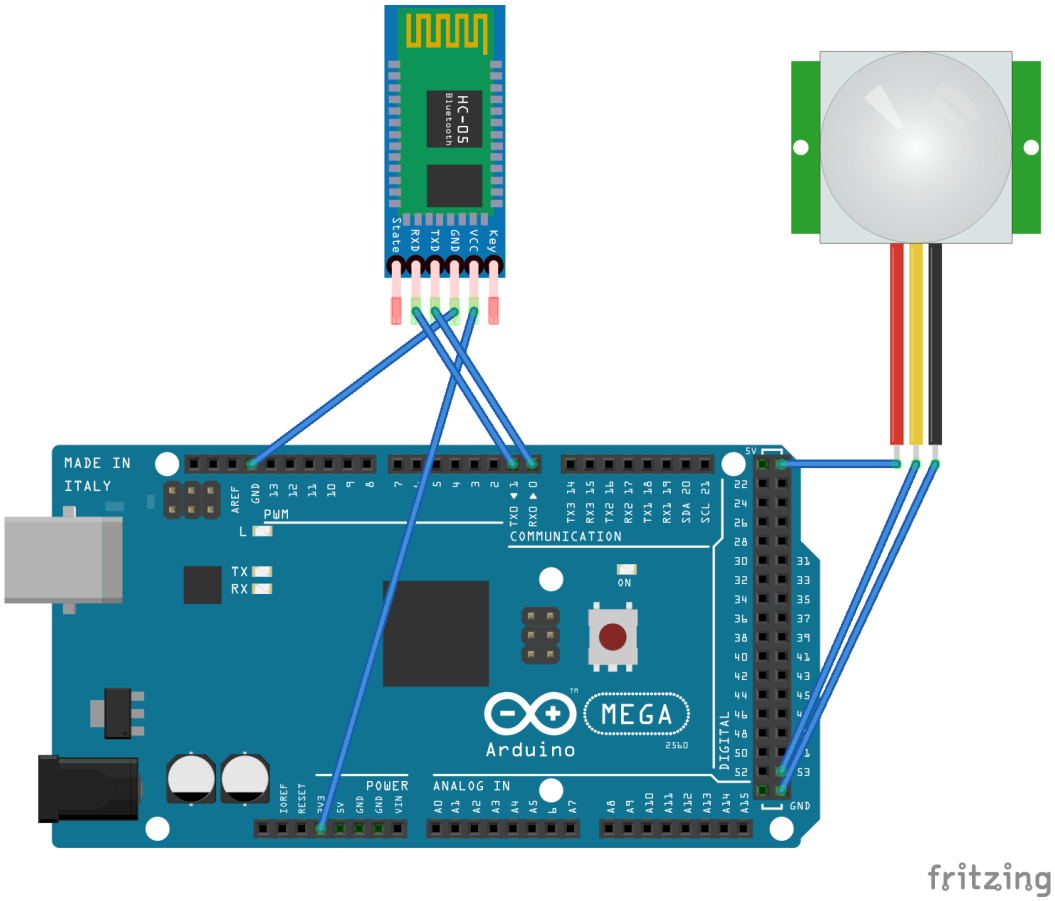
2 Testprotokol

Tests	Ergebnis
Bewegungungssensor erkennt bewegung	erfüllt
Bewegungen werden protokolliert	erfüllt
UserInput bis 4 Chars werden verarbeitet	erfüllt
Protokoll wird bei wenig speicher gestoppt	
Eingabe "list" gibt protokoll aus	erfüllt
Eingabe "help" gibt befehle aus	erfüllt
Eingabe "set" startet baseSetProzess	erfüllt
Eingabe "cler" leert liste	erfüllt
MemoryFunction verändert sich relativ manipulation des Logs	erfüllt
Über bluetooth ist Console verfügbar	erfüllt

3 Flussdiagram



4 Schema


HC-05
bluetooth
State
RXD
TXD
GND
VCC
Key
MADE IN
ITALY
PWM
COMMUNICATION
TX
RX
ON
MEGA
Arduino
2560
POWER
ANALOG IN
DIGITAL
5V
GND
fritzing

5 Reflexion

Ich fand es interessant mich mit Arduino zu befassen. Dieses Projekt ging jedoch nicht optimal voran. Ich hatte zu lange bei der Themenauswahl getrödelt und hatte gegen Ende sehr starke Zeitknappheit.

Dies hat jedoch nicht mit dem Unterricht zu tun, sondern eher, dass ich wenig Fantasy besitze, und mich auch nicht genug im arduino/c++ Bereich auskenne, um sagen zu können, was mir in welcher Zeit möglich ist. Dies hat mir recht zu schaffen gemacht, da ich Angst hatte ein zu kompliziertes Projekt anzugehen, welches mir die Zeit für die anderen Projekte fressen würde, oder etwas zu Simples, was nicht angemessen für einen Entwickler ist.
Diesem Projekt habe ich beim Terminplan meinen Optimalen Terminplan eingefügt. Diesen würde ich versuchen zu verfolgen, falls ich wieder in einer neuen Sprache ein Projekt mache und entspricht nicht dem echten Verlauf des Projektes.

6 3rd party resources:

Bluetoothsymbol:
https://www.google.com/imgres?imgurl=https%3A%2F%2Fupload.wikimedia.org%2Fwikipedia%2Fcommons%2Fthumb%2Fd%2Fda%2FBluetooth.svg%2F1200px-Bluetooth.svg.png&tbnid=_WZ2dDRLhTE7aM&vet=12ahUKEwis9-_q5uyEAxXRzQIHHb0WAt8QMygAegQIARBy..i&imgrefurl=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FList_of_Bluetooth_profiles&docid=r78QBA2axv113M&w=1200&h=1830&itg=1&q=bluetooth&client=firefox-b-d&ved=2ahUKEwis9-_q5uyEAxXRzQIHHb0WAt8QMygAegQIARBy

Bewegungssymbol:

https://www.google.com/imgres?imgurl=https%3A%2F%2Fcdn-icons-png.flaticon.com%2F512%2F1539%2F1539088.png&tbnid=B2Y70zSKhl6PYM&vet=12ahUKEwi8hvuE5-yEAxXi9QIHHXGcBkIQMygMegQIARB8..i&imgrefurl=https%3A%2F%2Fwww.flaticon.com%2Ffree-icon%2Fmotion-sensor_1539088&docid=dQ7G7QqdZRTdaM&w=512&h=512&q=movement%20sensor%20icon&client=firefox-b-d&ved=2ahUKEwi8hvuE5-yEAxXi9QIHHXGcBkIQMygMegQIARB8

Fritzing Motor Sensor: https://fritzing.org/projects/arduino-uno-motion-detector

Speichermessfunktion: https://forum.arduino.cc/t/how-to-create-and-free-dynamic-arrays-with-arduino/934662

7 Code

7.1 src.ino

```
#include <Arduino.h>
#include "SerialCommands.h"
#include "MemoryManagment.h"
#include "Logger.h"

// Author: simrem singh

const byte MOVMENT_SENSOR_PIN = 53;
const unsigned int MOVEMENT_ACTIVITY_TIME = 5000;

bool movementActive;
bool movementSensed;

unsigned long timeTillActivityDepricated;

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(MOVMENT_SENSOR_PIN, INPUT);
  // default values
  initLogger();

}

void loop()
{
  checkMovement();
  checkUserInput();
}

void checkMovement()
{
  movementSensed = digitalRead(MOVMENT_SENSOR_PIN) == HIGH;
  if (movementActive || movementSensed)
  { // movement sensor activity
    if ((!movementActive) && movementSensed)
    { // first contact
      Serial.println();
      Serial.print(F("Movement detected"));
      timeTillActivityDepricated = millis() + MOVEMENT_ACTIVITY_TIME;
      movementActive = true;

      createTimeStamp();
    }
    else if (movementActive && movementSensed)
```

```
    { // recurring contact (first contact not finished yet)
      timeTillActivityDepricated = millis() + MOVEMENT_ACTIVITY_TIME;
    }
    else// if (movementActive && (!movementSensed))
    { // check if contact finished
      movementActive = (timeTillActivityDepricated < millis());
      if (!movementActive)
      { // contact finished
        Serial.println();
        Serial.println(F("Movement stopped"));
      }
    }
  }
}

void checkUserInput()
{
  if (Serial.available() != 0)
  {
    interpret(getUserInput());
  }
}

void interpret(String userCommand)
{
  if (userCommand == "")
  {
    Serial.print(F("No Input."));
    Serial.println();
  }

  Serial.println();
  Serial.print(F("Command:"));
  userCommand.toUpperCase();
  userCommand.trim();
  Serial.print(userCommand);
  Serial.println();

  if (userCommand == "CLER" || userCommand == "CLR")
  {
    clearLog();
    return;
  }

  if (userCommand == "HELP" || userCommand == "?")
  {
    Serial.print(F("?,help = get Commands"));
    Serial.println();
    Serial.print(F("cler = reset log"));
    Serial.println();
```

```
    Serial.print(F("get,log,list = get log"));
    Serial.println();
    Serial.print(F("set = start set time sequence"));
    Serial.println();
    return;
  }
  if (userCommand == "GET" || userCommand == "LOG" || userCommand == "LIST")
  {
    PrintLog();
    return;
  }

  if (userCommand == "SET")
  {
    setBaseTime();
    return;
  }
  Serial.print(F("input not valid. Write help to get valid inputs"));
}
```

7.2 SerialCommands.h

```
#ifndef SERIALCOMMANDS_H //check if already defined/included in compiler
#define SERIALCOMMANDS_H //only define if not defined before

//Author: simrem singh

#include <Arduino.h>

String getUserInput()
{
  String command;
  byte streamlenght = 0;
  bool overflow = false; // end of Stream - prevent streamlenght overflow
  char inputChar;
  while (Serial.available() != 0)
  {
    streamlenght += 1;
    inputChar = Serial.read();
    delay(100);

    if (overflow)
    {
      Serial.print(inputChar);
    }
    else
    {
      overflow = (streamlenght > 4);
      if (!overflow)
```

```
      {
        command += inputChar;
      }
      else
      {
        Serial.print(F("Input above 4:"));
        Serial.print(inputChar);
      }
    }
  }
  return command;
}

#endif
```

7.3 MemoryManagment.h

```
#ifndef MEMORYMANAGEMENT_H //check if already defined/included in compiler
#define MEMORYMANAGEMENT_H //only define if not defined before

const unsigned int FreeMemoryLimit = 1000;
bool memoryLimitReached;

// Author: User "ec2021" on Arduino forum
// Original: https://forum.arduino.cc/t/how-to-create-and-free-dynamic-arrays-with-arduino/934662
// Formatted: https://forum.arduino.cc/t/how-to-create-and-free-dynamic-arrays-with-arduino/934662/12

extern unsigned int __bss_end;
extern void *__brkval;

int getFreeMemory()
{
  int free_memory;
  if ((int)__brkval == 0)
    free_memory = ((int)&free_memory) - ((int)&__bss_end);
  else
    free_memory = ((int)&free_memory) - ((int)__brkval);
  return free_memory;
}

#endif
```

7.4 Logger.h

```
#ifndef LOGGER_H // check if already defined/included in compiler
#define LOGGER_H // only define if not defined before
#include <Arduino.h>
#include "MemoryManagment.h"
```

```
#include "SerialCommands.h"

// Author: simrem singh

byte monthDays[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}; // Days in
each month



struct LogEntry
{
  byte day;
  byte month;
  // byte year;
  byte hour;
  byte minute;
  byte second;
  LogEntry *next; // linked list pointer
};

LogEntry *logListStack = NULL; // Head of the linked list
LogEntry baseValues;

void initLogger()
{
  baseValues.day = 10;
  baseValues.month = 3;
  baseValues.hour = 15;
  baseValues.minute = 5;
}

void printLogEntry(LogEntry *printEntry)
{
  Serial.println();
  Serial.print(printEntry->day);
  Serial.print(F("/"));
  Serial.print(printEntry->month);
  Serial.print(F("/-"));
  Serial.print(printEntry->hour);
  Serial.print(F(":"));
  Serial.print(printEntry->minute);
  Serial.print(F(":"));
  Serial.println(printEntry->second);
}

void PrintLog()
{
  Serial.println(F("Logged Entries:"));
  LogEntry *currentEntry = logListStack;
  while (currentEntry != NULL)
```

```
  {
    printLogEntry(currentEntry);
    currentEntry = currentEntry->next;
  }
  Serial.println();
}


void createTimeStamp()
{
  if (memoryLimitReached)
    return;

  byte tempMonth;
  byte tempDaysInMonth;
  unsigned long timeCalculation;

  LogEntry *newEntry = new LogEntry;
  timeCalculation = millis() / 1000; // millisends -> seconds
  newEntry->second = timeCalculation % 60;
  timeCalculation /= 60; // get seconds
  timeCalculation += baseValues.minute;
  newEntry->minute = timeCalculation % 60;
  timeCalculation /= 60; // get minutes
  timeCalculation += baseValues.hour;
  newEntry->hour = timeCalculation % 24;
  timeCalculation /= 24; // get days
  timeCalculation += baseValues.day;
  tempMonth = baseValues.month; // currentmonth
  tempDaysInMonth = monthDays[tempMonth - 1];
  while (tempDaysInMonth < timeCalculation)
  {
    timeCalculation -= tempDaysInMonth;
    tempMonth++;
    tempDaysInMonth = monthDays[tempMonth - 1];
  }
  newEntry->day = timeCalculation;
  newEntry->month = tempMonth;

  printLogEntry(newEntry);

  Serial.print(F("Free Memory:"));
  int freeMemory = getFreeMemory();
  Serial.print(String(freeMemory));
  Serial.println();
  memoryLimitReached = freeMemory < FreeMemoryLimit;
  if (memoryLimitReached)
    Serial.print(F("Memory Limit Reached. Logging stopped. Use command cler to
clear loglist!"));
```

```
  Serial.println();

  LogEntry *tempEntry = logListStack;
  logListStack = newEntry;
  logListStack->next = tempEntry;
}

void setBaseTime()
{
  byte tempStorage;

  Serial.print(F("month:"));
  while (!Serial.available())
  {
    delay(10);
  }
  tempStorage = getUserInput().toInt();
  if (tempStorage == 0 || tempStorage > 12)
  {
    Serial.println();
    Serial.print(F("invalid month."));
    return;
  }
  baseValues.month = tempStorage;

  Serial.print(F("day:"));
  while (!Serial.available())
  {
    delay(10);
  }
  tempStorage = getUserInput().toInt();
  if (tempStorage == 0 || tempStorage > monthDays[baseValues.month - 1])
  {
    Serial.println();
    Serial.print(F("invalid day. Max Day:"));
    Serial.print(String(monthDays[baseValues.month - 1]));
    return;
  }
  baseValues.day = tempStorage;

  Serial.print(F("hour:"));
  while (!Serial.available())
  {
    delay(10);
  }
  tempStorage = getUserInput().toInt();
  if (tempStorage > 23)
  {
    Serial.println();
    Serial.print(F("invalid hour. Max Hour:23"));
```

```
    return;
  }
  baseValues.hour = tempStorage;

  Serial.print(F("Minute:"));
  while (!Serial.available())
  {
    delay(10);
  }
  tempStorage = getUserInput().toInt();
  if (tempStorage > 59)
  {
    Serial.println();
    Serial.print(F("invalid hour. Max Hour:59"));
    return;
  }
  baseValues.minute = tempStorage;
  Serial.println();
  Serial.print(F("new base-time is set."));
  Serial.println();
}

void clearLog()
{
  Serial.println();
  memoryLimitReached = false;
  LogEntry *currentEntry = logListStack;
  LogEntry *nextEntry;
  while (currentEntry != NULL)
  {
    nextEntry = currentEntry->next;
    delete currentEntry;
    currentEntry = NULL;
    currentEntry = nextEntry;
  }
  logListStack = NULL;
  Serial.print(F("Log cleared."));
  Serial.println();
}

#endif
```