

1 Heim-Überwachung (5 Punkte + 3 Punkte)

1.1 Messung mit einem Server

Ändern Sie den Webserver von der Vorlesungs-Webseite foldenermaßen ab:

1. Messen Sie die aktuelle Temperatur, ein Sensor-Wert von 0 entspricht dabei einer Temperatur von -50°C und 410 entspricht 150°C . Benutzen Sie für die Umrechnung die Funktion `map()`.
2. Messen Sie den Zeitpunkt (in Sekunden seit Programmstart) der letzten Bewegung.
3. Der Bewegungssensor gibt für einen kurzen Zeitraum HIGH aus, wenn sich vor ihm etwas bewegt.
4. Stellen Sie die Temperatur und den Zeitpunkt der letzten Bewegung über den Webserver dar. (Das Grad-Zeichen “°” erreichen sie in HTML mit `°`.)

1.2 Speichern der Daten

Implementieren Sie eine Heim-Überwachung, die Messdaten über einen längeren Zeitraum in einer einfach verketteten Liste speichert:

1. Zeigen Sie die Temperatur der letzten 24 Stunden (im Stundentakt) auf Ihrem Webinterface an.
2. Zeigen Sie außerdem die Zeitpunkte der letzten 10 Bewegungen an. Dabei gilt eine Bewegung als *neu*, wenn sich seit mindestens 30 Minuten nichts bewegt hat.
3. **Hinweis:** Zum Testen dürfen (und sollten) Sie die Zeitintervalle verkürzen.

Einfach verkettete Liste

Auf dem letzten Blatt befand sich ein kleiner Fehler, was die Erstellung eines neuen Objektes betrifft. Auch wenn es in den Vorlesungsfolien richtig steht, scheint es keinem aufgefallen zu sein, obwohl der Code nicht einmal kompiliert. Dennoch hier eine korrigierte Version.

```
typedef struct queue_struct {
    int value;
    struct queue_struct *next;
} Queue;

Queue *head; //HEAD-Element, dass stets auf das erste Element zeigt

void addElement_head (Queue *newElem) {
    //Fuege Element an Anfang der Liste ein
    newElem->next = head;
    head = newElem;
}

Queue *newElem(int value) {
    //erzeuge neues Element und caste void-Pointer auf richtigen Typ
    Queue *newQueue = (Queue *)malloc(sizeof(Queue));

    newQueue->value = value;

    //default pointers
    newQueue->next = NULL;

    return newQueue;
}
```

2 Web-Display (4 Punkte + 2 Bonus)

In dieser Aufgabe sollen Sie einen Webserver auf dem Arduino implementieren, der Zeichenketten akzeptiert und diese auf dem LCD-Display anzeigt. Die Funktion `void handle_get(const char path[])` wird Ihnen bereits leer auf der Vorlesungs-Webseite zur Verfügung gestellt. Der Funktion wird die Zeichenkette übergeben, die den Pfad enthält, der vom Browser abgerufen wird. Um zu testen, was der Browser hier sendet, können Sie an dieser Stelle einen Aufruf zu `Serial.println()` einbauen. Erweitern Sie `handle_get()` folgendermaßen:

1. Wird der Pfad `/on` aufgerufen, soll die LCD-Hintergrundbeleuchtung eingeschaltet werden.
2. Wird der Pfad `/off` aufgerufen, soll die LCD-Hintergrundbeleuchtung ausgeschaltet werden.
3. Ansonsten soll der aufgerufene Pfad auf dem Display angezeigt werden.

2.1 Bonus

Ein URL-Pfad darf keine Leerzeichen und keine Sonderzeichen enthalten. Daher wird eine Zeichenkette wie "Hallo, Welt!" mit Hilfe von *URL-Kodierung* zu "Hallo,+Welt%21".¹

Erweitern Sie die Aufgabe so, dass der übergebene URL-kodierte Pfad dekodiert wird. Dazu müssen Sie:

1. Die Zeichenkette zeichenweise durchlaufen.
2. Das erste Zeichen, das immer ein `/` ist, ignorieren.
3. Wenn ein `+` gelesen wird, geben Sie ein Leerzeichen aus.
4. Wenn ein `%` gelesen wird, müssen die folgenden zwei Zeichen als Hexadezimal-Zahl eingelesen werden und dann das entsprechende Zeichen mithilfe von `void lcd.write(char c)` ausgegeben werden.
5. Alle anderen Zeichen sollen wie gegeben ausgegeben werden.
6. **Beispiel:** Wird `/Hallo,+Welt%21` aufgerufen, so soll `Hallo Welt!` angezeigt werden.
7. Zeichenketten, die länger als 16 Zeichen sind, sollen umgebrochen werden (an Zeichengrenzen, nicht an Wortgrenzen).

¹<http://de.wikipedia.org/wiki/URL-Encoding>