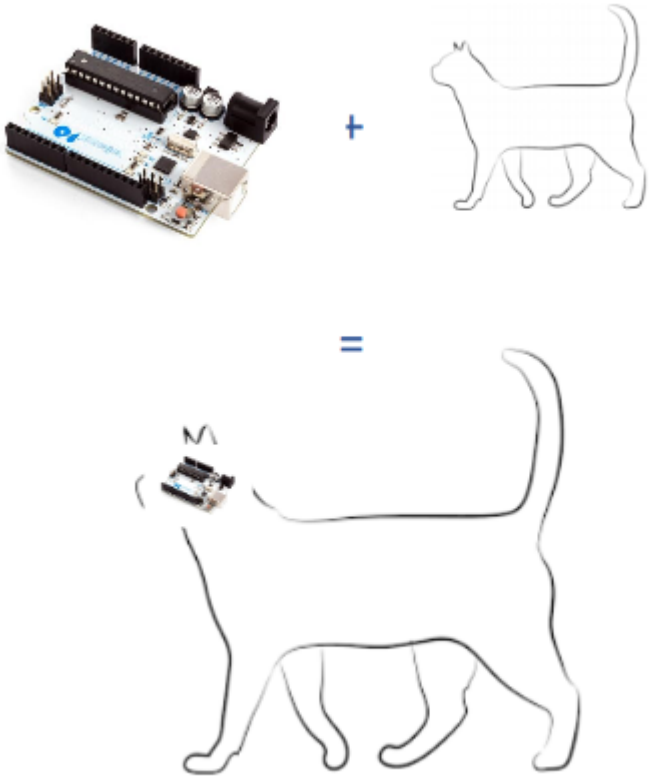
CoALA - Code A Little Animal

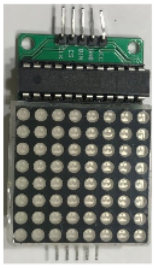
Workshop C

Sekundarstufe 1



Bildquelle: https://pixabay.com/de/katze-tier-die-silhouette-au%C3%9Ferhalb-1583459/License:CC0

Information: Die Liste der Materialien bezieht sich auf die Verwendung eines Arduino-Boards. Sie kann entweder von einem einzelnen Schüler, einem Team oder einer kleinen Gruppe von Schülern verwendet werden.

Material		
		
Arduinoboard	USB Kabel für den Arduino	Batterie (9,5 V) für den Arduino
		
Steckbrett	Verbindungskabel	PC*
		
8x8 LED Matrix (MAX-7219-based)	2 RGB LEDs	Lichtsensor
		
NFC Lesegerät RFID RC522	3 NFC Karten	Feuchtigkeitssensor

Materialien für die Tierkonstruktion (Pappkarton, ...)

*Bildquelle: https://pixabay.com/es/vectors/diseñador-gráfico-pc-macbook-mac-4562741/

WORKSHOP C

Sekundarstufe 1

Lerngebiet	Nummer	Arbeitsblatt	erledigt
Was du über den Arduino wissen solltest			
	1	Arduino-Board	☐
	2	Programmierung	☐
	3	Arduino-Oberfläche	☐
CoALA Programmiere ein kleines Tier			
	4	Dein Tier	☐
	5	Bedürfnisse deines Tieres	☐
	6	Vitalfunktionen deines Tieres	☐
	7	Zustand deines Tieres	☐
	8	Starr blickend	☐
	9	Berührung	☐
	10	Verkostung (1) - Essen	☐
	11	Verkostung (2) - Trinken	☐
	12	Körper deines Tieres	☐
	13	Leben deines Tieres	☐

1 Arbeitsblatt - Arduino

Mit dem Arduino stehen dir unzählige kreative Möglichkeiten zur Verfügung. Willst du einen Roboter bauen oder Nachrichten übermitteln? Mit einem Mikroprozessor wie den Arduino kannst du eine Reihe von Anweisungen (oder Programmen) erstellen, sodass der Arduino genau das tut, was du im Programm schreibst. Dies wird Programmierung genannt. Schaue dir deinen Arduino jetzt genau an.

Abbildung 1: Arduino UNO R3 Bord

Ein Arduino basiert auf einem Mikrocontroller (ein elektronischer Prozessor, der ein einzelnes Programm ausführt). Er hat die folgenden Anschlussports:

- 14 digitale Pins (ein/aus), die als Ein- oder Ausgänge konfiguriert werden können. Von diesen können 6 Ausgänge als PWM-Ausgänge (~) konfiguriert werden.
- 6 analoge Eingangspins
- 1 USB-Anschluss
- 1 externer Stromanschluss (maximal 12 V)
- Weitere Informationen findest du unter: `https://www.arduino.cc/`.

1. Aufgabe

Identifiziere auf der Arduino-Platine die Hauptkomponenten, die auf der Platine integriert sind, und finde heraus, was sie sind und welche Funktion sie haben.

2. Aufgabe

Identifiziere auf der Arduino-Platine die verschiedenen Anschlussports/Pins und gebe an, was die Begriffe „digital“, „analog“ und „PMW“ bedeuten.

2 Arbeitsblatt - Programmierung

Mit dem Arduino wirst du in der Lage sein, Programme zur Erstellung digitaler Projekte zu schreiben. Der Zweck eines Programms ist es, eine Folge von Anweisungen auszuführen, die eine bestimmte Aufgabe auf einem Computer automatisiert. Das Schreiben dieser Befehlssequenz wird als Codierung bezeichnet. Die Anweisungen werden in einer so genannten Programmiersprache geschrieben. Der Arduino verwendet eine eigene Programmiersprache und eine eigene integrierte Entwicklungsumgebung (Arduino IDE). Die Arduino-Software (IDE) ermöglicht es dir, Programme zu schreiben und sie auf dein Board hochzuladen.

2.1 Arduino IDE

Auf der Arduino-Seite (`https://www.arduino.cc/en/Guide/HomePage`) findest du zwei Möglichkeiten:

1. Online-IDE (Arduino Web-Editor): Sie ermöglicht es dir, deine Skizzen in der Cloud zu speichern, sie von jedem Gerät aus verfügbar zu haben und ein Backup zu erstellen. So wirst du immer die aktuellste Version der IDE haben, ohne dass du Updates oder von der Gemeinschaft erstellte Bibliotheken installieren musst. Sie ist verfügbar unter: `https://create.arduino.cc/`. Die Online-IDE erfordert die Erstellung eines Benutzerkontos.

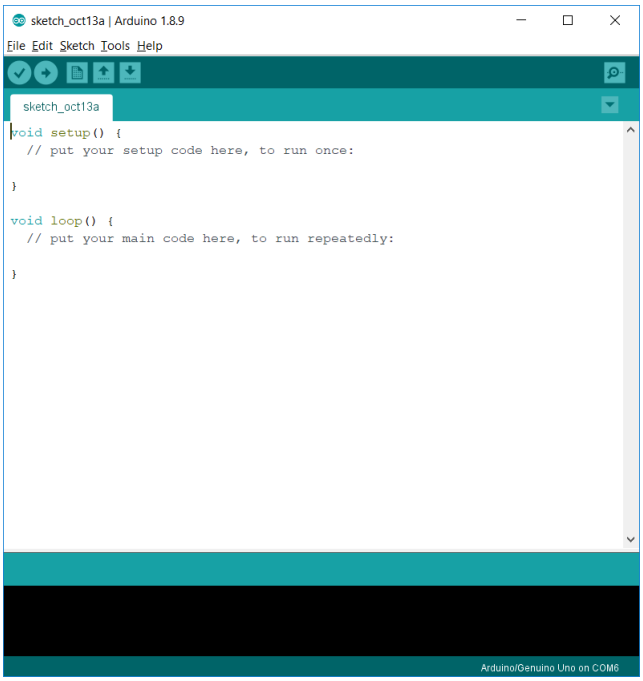


Abbildung 2: Online IDE (Arduino Web Editor)

2. Desktop-IDE: Sie wird auf deinem eigenen Computer installiert, sodass du offline arbeiten kannst. Du kannst die neueste Version auf der Seite `https://www.arduino.cc/en/Guide/HomePage` erhalten. Sobald die IDE installiert ist, kannst du über das Symbol darauf zugreifen:

2.2 Arduino-Programme

Wenn der Arduino-Editor geöffnet wird, erscheint der folgende Bildschirm:

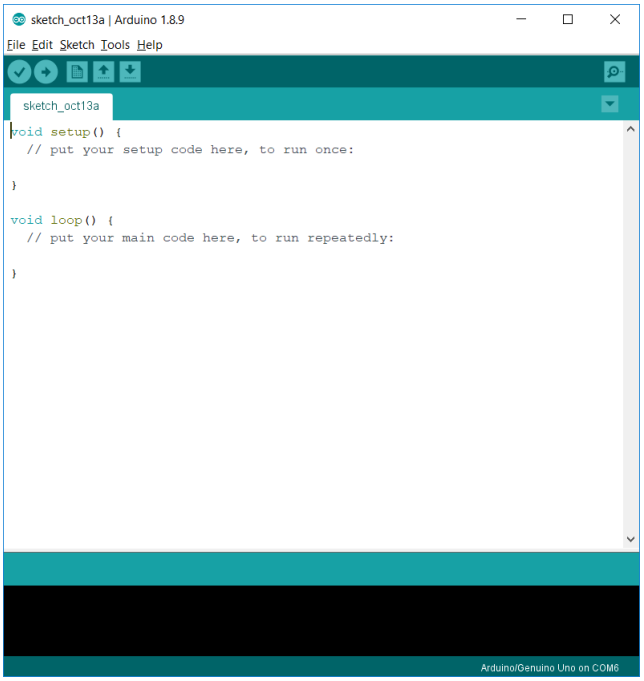


Abbildung 3: Arduino Programmstruktur

Der Bildschirm zeigt bereits das Arduino-Programmfenster. Ein Arduino-Programm (auch Skizze genannt) hat zwei Hauptteile: die Setup-Funktion und die Schleifenfunktion (loop). Die Setup-Funktion dient zur Initialisierung von Variablen, Pin-Modi, zum Starten der Verwendung von Bibliotheken usw. Der Setup-Abschnitt wird nur einmal ausgeführt, nach jedem Einschalten oder Reset der Arduino-Platine.

Die Schleifenfunktion macht genau das, was ihr Name vermuten lässt: führt Schleifen hintereinander aus, so dass dein Programm sich ändern und reagieren kann. Kommentare sind Zeilen im Programm, die dazu dienen, sich selbst oder andere über die Funktionsweise des Programms zu informieren. Ein einzeiliger Kommentar wird „// Kommentar“, während ein mehrzeiliger Kommentar „/* Kommentar #1 ... */“ geschrieben wird.

1. Aufgabe

Programmiere das Arduino-Board. Führe die folgenden Schritte aus, um das Board zu programmieren:

1. Verbinde das Arduino-Board über ein USB-Kabel mit dem Computer.
2. Installiere die Treiber (`https://www.arduino.cc/en/Guide/HomePage`) auf dem Arduino-Board, falls erforderlich
3. Starte den Arduino Web-Editor oder die Desktop-IDE
4. Öffne ein Beispiel: Blinken (die integrierte LED auf dem Arduino-Board blinkt).
5. Wähle die Art der Tafel.
6. Wähle ggf. den seriellen Kommunikationsanschluss aus.
7. Lade das Programm auf das Arduino-Board hoch.

Abbildung 4: Öffnen eines Arduino Programms

3 Arbeitsblatt - Arduino-Schnittstellen

Wie wir in den vorangegangenen Arbeitsblättern gesehen haben, ist das Arduino-Board ein recht leistungsfähiges Gerät zur Steuerung und Ausführung komplexer Aufgaben nach kodierten Anweisungen. Allerdings ist die Arduino-Platine an sich begrenzt, da die Arduino-Schnittstellen nur elektrische Signale an den Pins auf der Platine sind. Daher werden zur Anbindung an die reale Welt einige zusätzliche Elemente benötigt: Sensoren und Aktoren.

Ein Sensor ist ein Gerät, das physikalische Eigenschaften der Umgebung erfasst und die Informationen an andere Elektronik, häufig einen Prozessor, sendet, wie z.B.: Spannungen, Ströme, Temperatur, Licht, Schall, ...

Ein Aktor ist eine Komponente eines Systems, die für die Bewegung und/oder Steuerung eines Mechanismus oder einer Maschine verantwortlich ist.

Diese Sensoren und Aktoren sind in der Regel so vorbereitet, dass sie über die Pins auf der Platine mit der Arduino-Platine verbunden werden bzw. mit ihr kommunizieren können. Wie wir in Arbeitsblatt 1 gesehen haben, sind diese Pins von verschiedenen Typen, digital und analog.

3.1 Digitale Eingänge/Ausgänge

Öffne das Programa_1 im Editor. Du siehst die beiden Abschnitte setup() und loop(). Schließe den Arduino wie

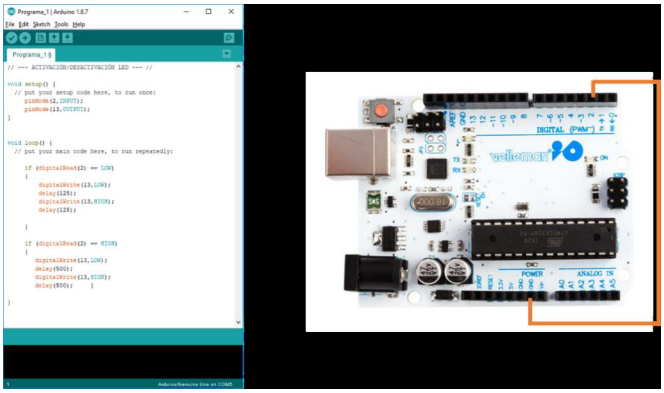


Abbildung 5: Digitale Eingänge/Ausgänge

in der Abbildung gezeigt an (mit einem Kabel, das Pin 2 und den GND-Pin verbindet) und lade Programa_1 in das Arduino-Board.

1. Aufgabe

Analysiere die Funktionsweise des Programms. Das Arduino-Board hat eine integrierte LED, die mit Pin 13 verbunden ist. Warum schaltet sich die LED auf dem Board an und aus? Was macht deiner Meinung nach das Programm in den einzelnen Abschnitten setup() und loop() und was macht jede der Anweisungen?

---

2. Aufgabe

Wenn du das Programm analysiert hast, wie müsstest du das Kabel anschließen, um die Blinkfrequenz zu ändern? Welche Änderungen müsstest du im Programm vornehmen, damit das Blinken langsamer wird? Und schneller?

---

3. Aufgabe

Kannst du es mit Pin 3 statt Pin 2 zum Funktionieren bringen? Und kannst du das Blinken mit vier Frequenzen machen? (Hinweis: Es werden zwei Drähte benötigt).

3.2 Analoge Eingänge

Öffne das **program_2a** im Arduino-Editor. Wir arbeiten jetzt mit analogen Eingängen, d.h. sie können kontinuierlich zwischen 0V und 5V variieren. Dazu werden wir mit einem Umgebungslichtsensor arbeiten. Entferne die Arduino-Platine, um die Anschlüsse zu ändern. Schließe den Arduino an den Sensor an, wie in der nächsten

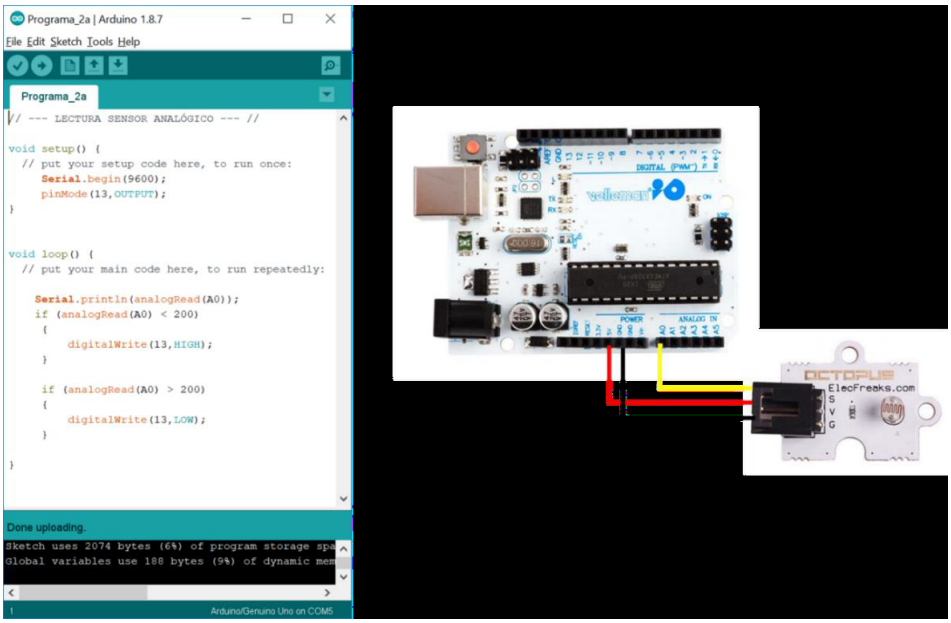


Abbildung 6: Analoge Eingänge

Abbildung gezeigt: Verbinde schwarzes Kabel (Ground) mit GND sowie das rote Kabel (Stromversorgung) mit 5V und analoges (Sensorausgangs-) Signal an Eingangspin A0. Lade das Programm hoch.

1. Aufgabe

Analysiere das Programm. Warum schaltet sich die LED ein und aus?

---

2. Aufgabe

Aktiviere den „Seriellen Monitor“ in der Registerkarte „Werkzeuge“ und schaue dir die erfassten Maximal- und Minimalwerte an. Beobachte, wann LED in Bezug auf den Wert des „Serial Monitor“ an und aus geht. Wie kannst du den Belichtungsmesser mehr oder weniger empfindlich machen?

3.3 Analoge Ausgänge (PWM)

Öffne das **program_2b** im Arduino-Editor. Wir arbeiten jetzt mit den analogen Ausgängen, d.h. sie können kontinuierlich zwischen 0V und 5V variieren. Dies wird durch die Erzeugung einer Impulsfolge mit variabler Breite namens PWM erreicht. Der Ausgang ist digital, aber er variiert so schnell, dass der Mittelwert jeden Wert zwischen 0 und 5V annehmen kann.

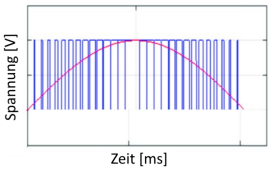


Abbildung 7: PWM Signal

Dazu werden wir mit dem Umgebungslichtsensor arbeiten, den wir zur Steuerung der Helligkeit einer LED verwenden werden. Schließe den Arduino an den Sensor an, wie in der nächsten Abbildung gezeigt: Schließe das schwarzes Kabel (ground) an GND, das rote (Stromversorgung) an 5V und analoges (Sensorausgangs-) Signal an Eingangspin A0 und die LED und einen 330Ω Serienwiderstand an Pin 13 an. Lade das Programm auf die Platine.

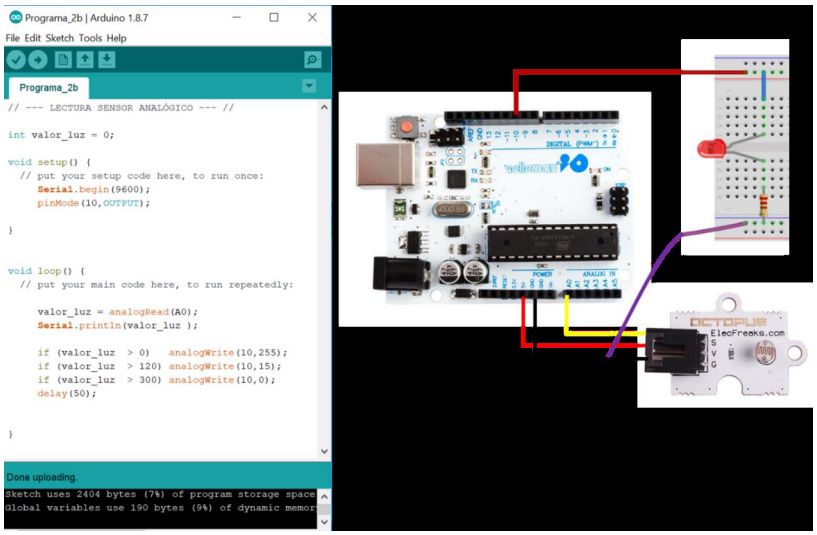


Abbildung 8: Analoge Ausgänge

1. Aufgabe

Analysiere das Programm. Wie viele Helligkeitsstufen haben wir programmiert? Wie könnten wir weitere Stufen hinzufügen?

2. Aufgabe

Wie würdest du das Programm abändern, sodass die Helligkeit der LED proportional zur Lichtintensität ist?

Anmerkung: Um die Verbindungen zu testen, verwenden wir ein Lochrasterplatte, indem wir die Arduino-Pins 5V (Versorgung) und GND (Masse) mit der Lochrasterplatte verbinden, wie in der Abbildung gezeigt. Auf diese Weise können wir Tests einfacher durchführen (um jeden analogen Sensor anzuschließen, benötigen wir eine 5V- und eine GND-Verbindung).

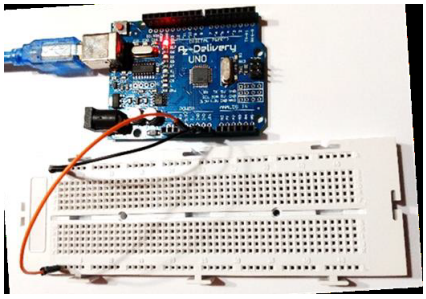


Abbildung 9: Lochrasterplatte

4 Arbeitsblatt - Dein Tier

Ein Haustier (Hund, Katze, Kaninchen, ...) ist ein domestiziertes Tier, das den Zweck hat, Gesellschaft zu leisten.

1. Aufgabe

Wähle ein Haustier aus.

Spreche mit deiner Lehrkraft über deine Wahl. Notiere dir das von dir mit deiner Lehrkraft gewählte Haustier:

_____

2. Aufgabe

Schreibe ein Profil zu deinem Tier.

Allgemeines
- Name: _____
- Ordnung: _____
- Merkmale: _____
- Komforttemperatur: _____
- Lebensdauer: _____

Physikalische Eigenschaften
- Körpergröße: _____
- Körperbreite: _____
- Gewicht: _____

Lebensstil
- Diät (Essen und Getränke): _____
- Aktivität (viel oder wenig): _____
- Schlaf (viel oder wenig): _____

5 Arbeitsblatt - Bedürfnisse deines Haustiers

Durch das Schreiben des Profils hast du bereits eine Menge über dein Haustier erfahren. Bevor du jedoch in Erwägung ziehst, dir ein Haustier anzuschaffen, ist es wichtig zu wissen, was das Haustier braucht. Ein Haustier hat, wie jedes andere Lebewesen auch, verschiedene Lebensfunktionen. Eine Lebensfunktion ist in der Biologie einer der drei Prozesse oder Funktionen, die alle Lebewesen ausführen: Ernährung (einschließlich Atmung), Interaktion (Beziehung) und Fortpflanzung.

1. Aufgabe

Was braucht dein Haustier? Schreibe dir die fünf wichtigsten Punkte auf.

Bald wirst du ein Haustier haben können (über das Arduino-Board). Du wirst dein Haustier erstellen und richtig pflegen können. Du wirst simulieren, wie sich dein Haustier „fühlt“, indem du Sensoren und Aktoren verwendest und das Verhalten deines Haustiers auf den nächsten Arbeitsblättern programmierst.

2. Aufgabe

Wie machst du dein Haustier glücklich? Überlege dir, wie du das Haustier pflegen oder aufziehen kannst, wie der Zustand oder die Gefühle des Haustiers sein könnten und wie sie durch das Verhalten des Besitzers verändert werden könnten.

6 Arbeitsblatt - Vitalfunktionen deines Haustiers

Mit Hilfe elektronischer Geräte wollen wir einige der lebenswichtigen Funktionen des Haustiers simulieren (im Wesentlichen Ernährungs- und Beziehungsfunktionen). Die zu simulierenden Funktionen sind:

- Ernährung → Essen / Trinken
- Beziehung → Blick, Streicheleinheiten, ...

In diesem Arbeitsblatt werden wir unter den verschiedenen Arten von Sensoren und Aktoren, die zur Verfügung stehen, einige auswählen, um diese Funktionen zu simulieren.

1. Aufgabe

1.1 Wie kannst du wissen, wie sich dein Haustier fühlt?

1.2 Wie kannst du überprüfen, ob das Zimmer für dein Haustier warm genug ist?

1.3 Wie kannst du überprüfen, ob dein Haustier gestreichelt werden will?

1.4 Du gibst deinem Haustier zwei verschiedene Nahrungsmittel. Woher weißt du, was dein Haustier bevorzugt?

1.5 Wie kannst du wissen, ob dein Haustier durstig ist?

7 Arbeitsblatt - Zustand deines Haustiers

Um den physischen und emotionalen Zustand deines Haustiers zu visualisieren, verwenden wir eine 8x8-LED-Matrix, die von einem MAX7219-Gerät gesteuert wird.

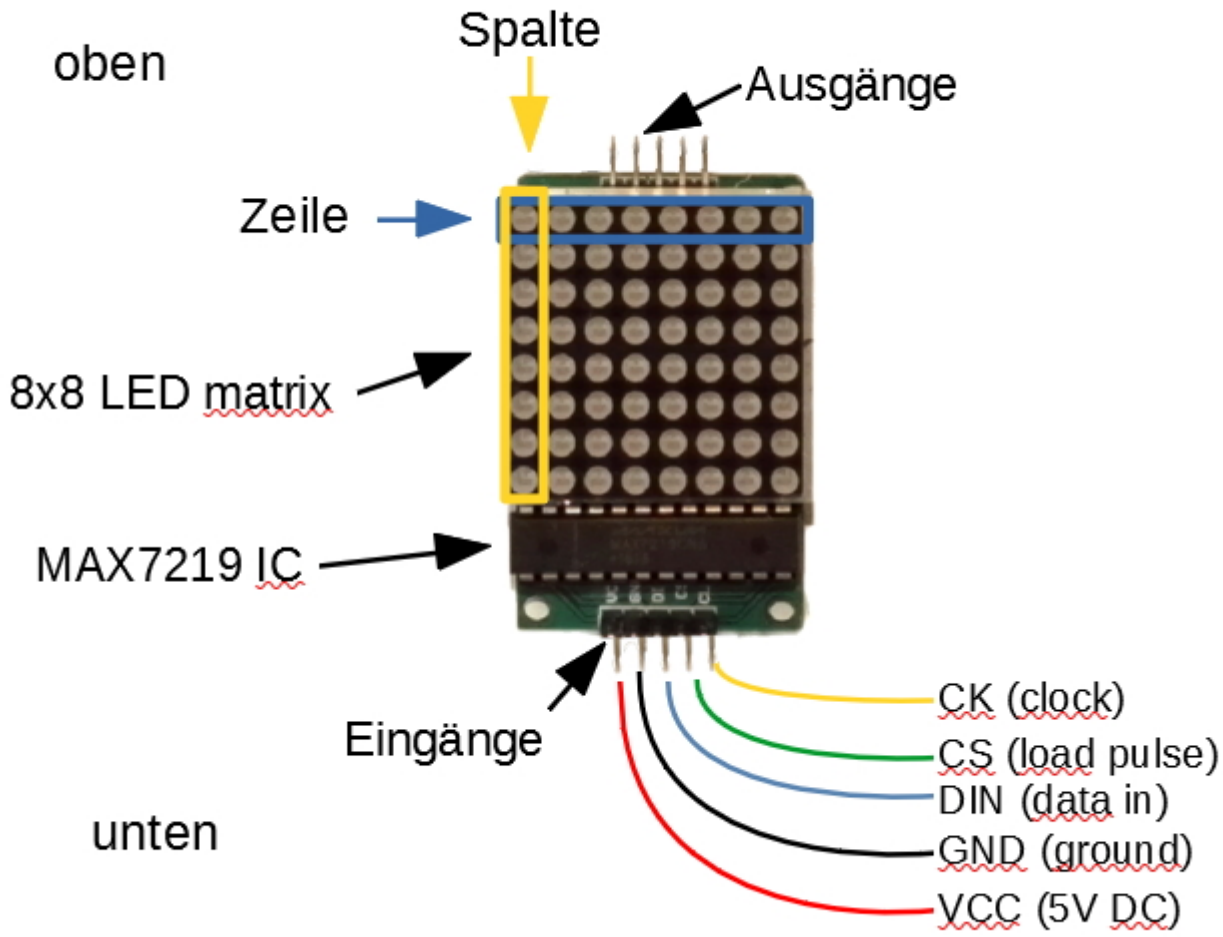


Abbildung 10: 8x8 LEDs Matrix

Die Verbindung zwischen dem MAX7219 und dem Arduino erfolgt über einen Kommunikationsbus mit drei Drähten (Clock, CS und DI). Das Anschlussschema ist in der nächsten Abbildung dargestellt:

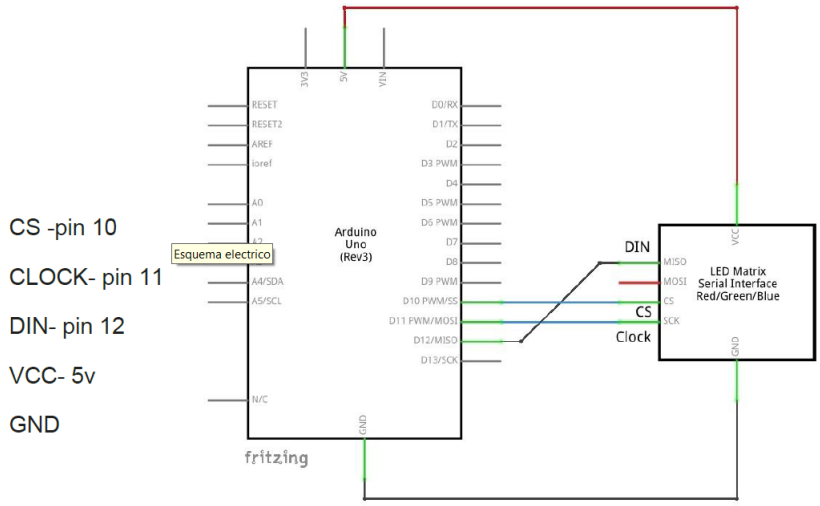
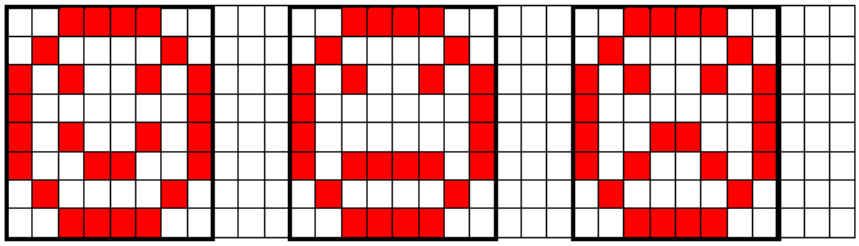


Abbildung 11: MAX7219 - Arduino Oberfläche

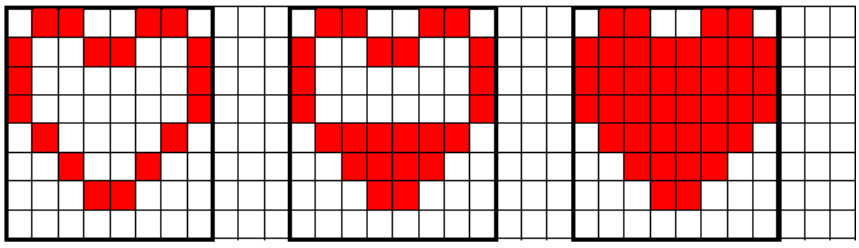
1. Aufgabe

Schließe die LED-Matrix wie angegeben an und installiere die Bibliothek (LedControl.h) zur Verwaltung der LEDs gemäß den Anweisungen unter
`https://www.instructables.com/id/LED-Matrix-with-Arduino/`.
Sobald die Bibliothek installiert ist, öffne den Arduino-Editor und im Menü Datei/Beispiel/LedControl/ finde verschiedene Beispiele zum Verständnis der Programmierung. Öffne das Programm Programa_3a, das verschiedene Gesichter in der LED-Matrix zeigt, und lade es hoch:



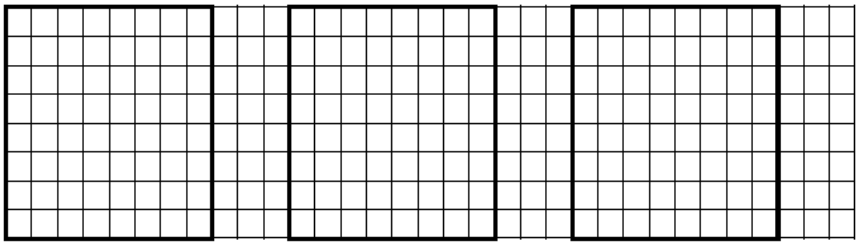
2. Aufgabe

Modifiziere das Programm so, dass es ein volles Herz zeichnet, das sich leert. Verwende die quadratische Vorlage. Speichere sie unter dem Namen Programa_3b.



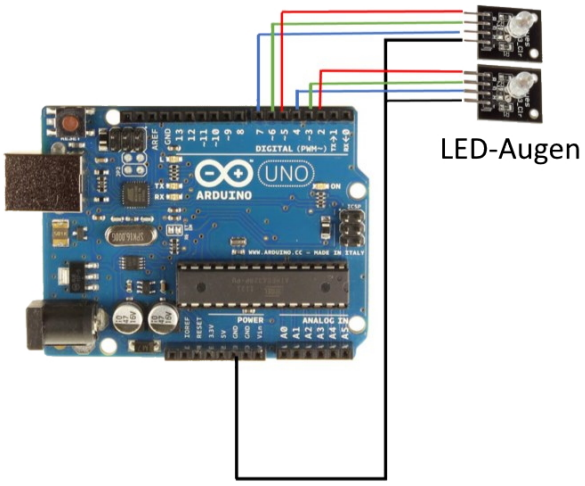
3. Aufgabe

Verändere das Programm so, dass du auf dem Bildschirm verschiedene Arten von Futtermitteln zeichnest, die dein Haustier mag und die dein Haustier nicht mag. Speichere das Programm unter dem Namen Programa_3c.



8 Arbeitsblatt - Vertiefung

Eine weitere Möglichkeit, die Stimmung deines Haustiers zu zeigen, ist die Verwendung von RGB-LEDs, die das Betrachten simulieren und so programmiert sind, dass sie je nach Zustand die Farbe wechseln.



Es erzeugt drei verschiedene Programme, eines, bei dem die LEDs grün sind (das Haustier fühlt sich gut ••) mit dem Namen **Program_4a**, ein anderes, bei dem die LEDs blau sind (das Haustier ändert seinen Status ••) mit dem Namen **Program_4b** und ein weiteres mit roten Augen (das Haustier fühlt sich schlecht ••) mit dem Namen **Program_4c**.

Sobald die Pins als Ausgang angezeigt werden: pinMode(2, OUTPUT) in der setup()-Funktion; es wird in der loop()-Funktion angezeigt; welche Pins in jedem Programm aktiv sind oder nicht:

digitalWrite(2, LOW); oder digitalWrite(3, HIGH);

Wenn du farbige Drähte verwendest, ist es viel einfacher zu wissen, was jedem Draht entspricht.

Du kannst auch versuchen, einen Summer zu programmieren (dein Haustier wird sich beschweren, wenn es sich nicht wohl fühlt) oder einen anderen Aktor mit dem gleichen Zweck. Programmieren_5. Simuliere verschiedene Geräusche und raten deiner Lehrkraft, einen davon auszuwählen.

Aktuator Buzzer
- GND
Pin 8
+ 5V

Beantworte die folgenden Fragen:

1. Frage Sind die Geräte, die du benutzt hast, Ein- oder Ausgänge?

RGB-LED:

Summer-Sensor:

---

2. Frage Sind sie digital oder analog?

RGB-LED:

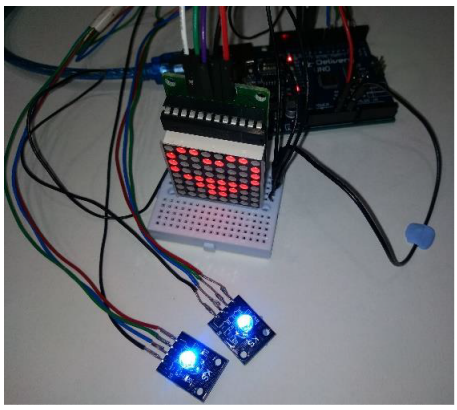
Summer-Sensor:

Begründe deine Antworten.

Du kannst mehr als einen Sensor verwenden, um anzuzeigen, wie sich dein Haustier fühlt:

1. Aufgabe

Du kannst abstimmen, was in der LED-Matrix mit den RGB-LEDs passiert, z.B. wenn die Augen die Farbe wechseln, von grün über blau bis hin zu rot. **program_6**. (Kombiniere **program_3c** mit **program_4**). Da du viele Kabel anschließen musst, verwende eine Lochrasterplatine.



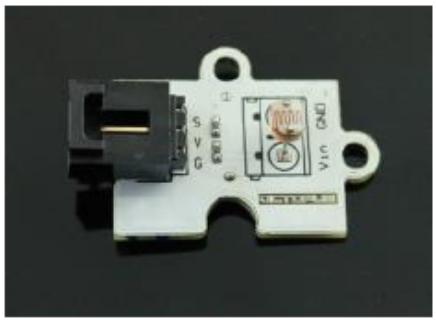
2. Aufgabe

Du kannst den Summer-Sensor, mit dem was in der LED-Matrix passiert, verbinden und das Haustier könnte so einen Ton erzeugen, wenn sein Herz fast leer ist. **program_7**.

9 Arbeitsblatt - Streicheln

Dein Haustier will gestreichelt werden, sonst fühlt es sich einsam. Verwende den Lichtsensor, um diese Aktion zu simulieren. Wenn das Licht den Sensor nicht erreicht (verdeckter Sensor), weiß das Haustier, dass es gestreichelt wird. Wenn eine Weile vergeht, ohne dass es gestreichelt wird, fühlt sich dein Haustier traurig und sendet uns ein Signal: Das Herz beginnt sich zu leeren, und die Augen ändern ihre Farbe.

Da der Lichtsensor ein analoger Sensor ist, muss er an GND (schwarzes G-Kabel), 5V (rotes V-Kabel) und das Ausgangssignal S an einen analogen Eingangspin angeschlossen werden (in unserem Fall verwenden wir A0). Siehe **program_2a**.



Lichtsensor
S/S - A0
V/ - 5V
G/- - GND

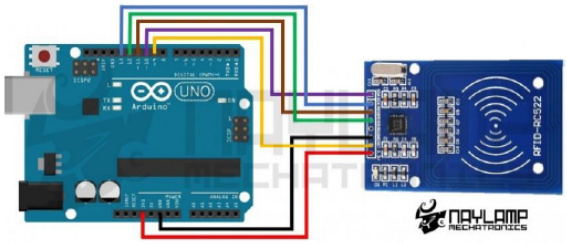
1. Aufgabe

Schließe den Lichtsensor wie angegeben an (ohne das LED-Array oder die RGB-LEDs zu trennen). Öffne dein gespeichertes Programm, **program_6**, und füge den Lichtsensorcode hinzu, um das Herz zu füllen, wenn der Sensor abgedeckt ist. Speichere es als **program_8**. Das Herz leert sich, wenn wir das Tier nicht berühren. Programmiere dieses Verhalten mit einer if-Funktion:

```
if (analogRead (A0)> 200) { heart_stat = 5; }
```

10 Arbeitsblatt - Nahrung (Lebensmittel)

Dein Haustier braucht Nahrung zum Überleben. Wir werden die verschiedenen Arten von Futter mit einem drahtlosen NFC-Kartenleser simulieren. In unserem Fall werden wir ein RFID-Lesegerät RC522 verwenden.



Lesegerät RFID RC522
RST - Pin 9
SDA(SS) - Pin 10
MOSI - Pin 11
MISO - Pin 12
SCK - PIN 13
GND - GND
3,3V - 3,3V

Damit das RFID-Lesegerät RC522 ordnungsgemäß funktioniert, muss die Bibliothek „MFRC522.h“ heruntergeladen und installiert werden, wobei die Anweisungen von
`https://naylampmechatronics.com/blog/22_Tutorial-Lector-RFID-RC522.html` zu befolgen sind.

Sobald die Bibliothek heruntergeladen ist, öffne den Arduino-Editor und im Menü Datei/Beispiele/LedControl/ kannst du verschiedene Programme ausprobieren, um zu sehen, wie es funktioniert.

1. Aufgabe

1.1

Schließe den NFC-Leser an eine Arduino-Platine an, wie in der vorigen Abbildung gezeigt, und führe das Beispielprogramm rfid_read_personal_data aus, öffne den „Serial Monitor“ aus dem Menü „Tools“, um den Inhalt jedes NFC-Chips lesen zu können. Bitte die Lehrkraft um die vorbereiteten Steckplatine und zeichne auf jede Steckplatine ein entsprechendes Bild für die jeweilige Textnachricht. Benutze einen nicht permanenten Marker!

1.2

Du kannst auf die Karten oder Schlüsselanhänger schreiben, was du für dein Haustier für angemessen hältst. (Denke daran, dass wir auf Blatt 7A bereits 3 Nahrungsmittel zeichnen. **program_3c**). Installiere nun das Beispielprogramm rfid_write_personal_data, öffne den „Serial Monitor“ aus dem Menü „Tools“, um den Inhalt jeder Karte oder jedes Schlüsselanhängers zu schreiben.

2. Aufgabe

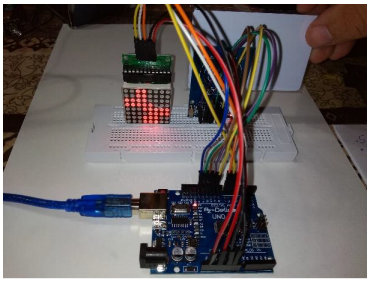
2.1

Lade das **program_9** hoch und überprüfe, wie es funktioniert.

2.2

Da wir nicht genügend digitale Pins haben, werden wir die gleichen Pins für die beiden RGB-LEDs (Pin 2, Pin 3 und Pin 4) verwenden, wir schließen sie parallel an. Und wir werden die Pins 5, 6, 7 zum Anschluss der LEDS-Matrix verwenden. Nach der Änderung der Verbindungen modifiziere **program_8** und speichere es als **program_8b**. Prüfe, ob alles funktioniert.

2.3

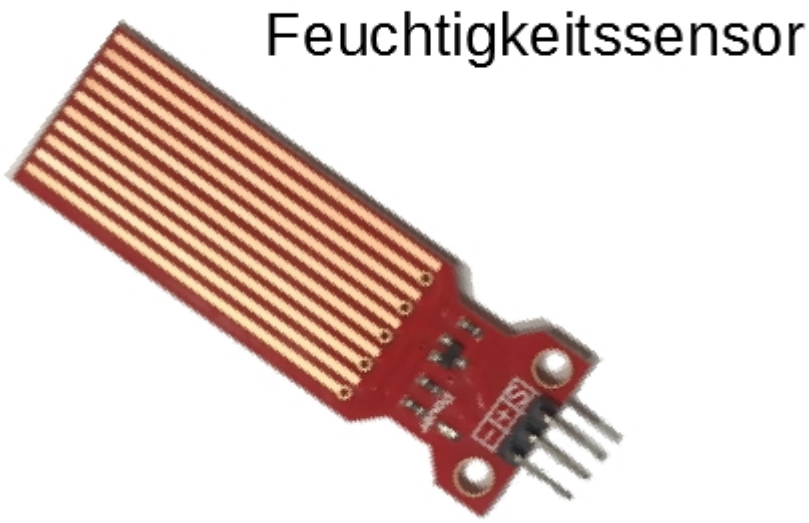


Füge den NFC-Leser zum Gerät deines Haustiers hinzu. Wir müssen das Programm so modifizieren, dass es glücklich wird, wenn wir ihm das Futter geben, das es gerne isst (d.h. das Herz füllt sich). Zeige das Programm deiner Lehrkraft.

Um es visueller zu machen, können wir dieses Programm mit dem **program_3c** zusammenfügen, in dem wir das Futter, das das NFC-Lesegerät liest, in die LED-Matrix zeichnen. Speichere es als **program_10**.

11 Arbeitsblatt - Durst (Getränke)

Dein Haustier muss auch trinken (Wasser), um zu überleben. Wir werden einen Feuchtigkeitssensor verwenden, um die Zunge des Haustiers simulieren. Verwende einen Feuchtigkeitssensor, um zu messen, ob dein Haustier genug Wasser hat. Abhängig von der Luftfeuchtigkeit bietet es Werte zwischen 0 und 800.



1. Aufgabe

Schließe den Feuchtigkeitssensor an eine Arduino-Platine an wie in der vorherige Abbildung und verwende ein Glas Wasser, um den Ausgabewert zu bestimmen:

- wenn der Sensor nicht in Wasser eingetaucht ist.

  Wert: _____

- wenn die Hälfte des Sensors ins Wasser eingetaucht ist.

  Wert: _____

- wenn der Sensor vollständig ins Wasser eingetaucht ist.

  Wert: _____

Achtung: Achte darauf, dass der Sensor nicht über die elektrischen Kontakte hinaus eingetaucht wird.

2. Aufgabe

Programmiere eine Sequenz, in der das Herz gefüllt wird, wenn man bedenkt, dass die Werte des Sensormesswertes angemessen ist. Speichere das Programm als **program_11**. Zeige dein Programm deiner Lehrkraft.

12 Arbeitsblatt - Körper deines Haustiers

In diesem Arbeitsblatt werden wir den Körper des Haustiers unter Verwendung einiger Vorlagen von Tieren wie Hund, Katze, ... gestalten.

1. Aufgabe

1.1

Du benötigst die richtige Vorlage für dein Haustier (Vorlagen 4-11). Schneide den Körper deines Haustiers aus und klebe ihn auf einen Karton. Lass auf der Oberseite der Karte genügend Platz für den Arduino.

1.2

Schneide mit einem Cutter entlang der roten Linien (nicht an den roten Flächen) aus. Lass dir von deiner Lehrkraft helfen, wenn es irgendwelche Schwierigkeiten gibt.

1.3

Klebe die verschiedenen Geräte in die entsprechenden Bereiche deines Haustiers.

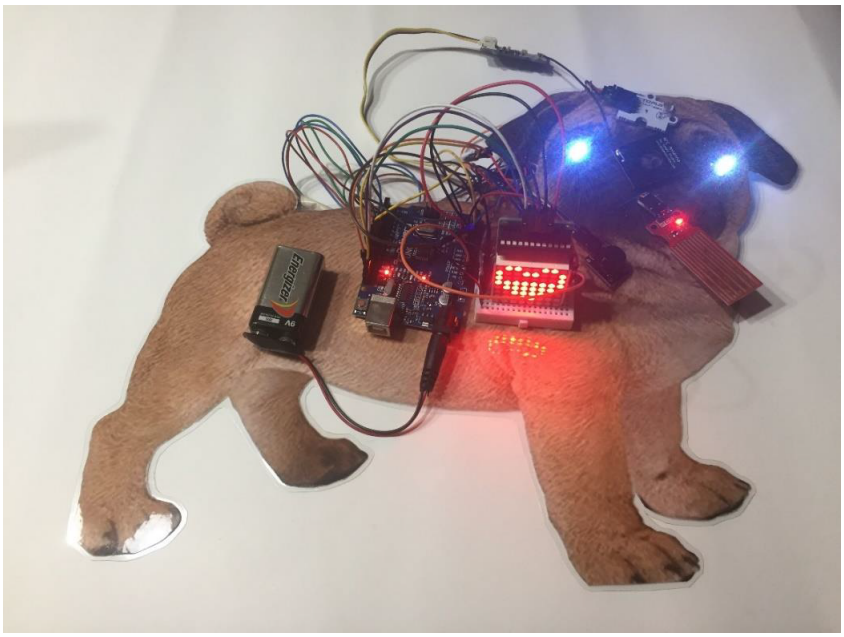
1.4

Halte das Arduino-Board über das Haustier. Lass etwas Platz.

1.5

Lege die Zunge, die Augen, das NFC-Lesegerät und den Lichtsensor auf den Kopf deines Haustiers. Lege das Herz auf den Körper deines Haustiers. Schließe die Geräte an das Arduino-Board an.

Achtung: Jedes Kabel muss an der richtigen Stelle angebracht werden. Überprüfe die Platine auf Anweisungen.



13 Arbeitsblatt - das Leben deines Haustiers

Wenn alles an seinem Platz ist, wollen wir das Verhalten deines Haustiers überprüfen. Schalte deinen Arduino ein.

1. Aufgabe

1.1

Reagiert dein Haustier?

1.2

Berühre dein Haustier.

1.3

Füttere dein Haustier.

1.4

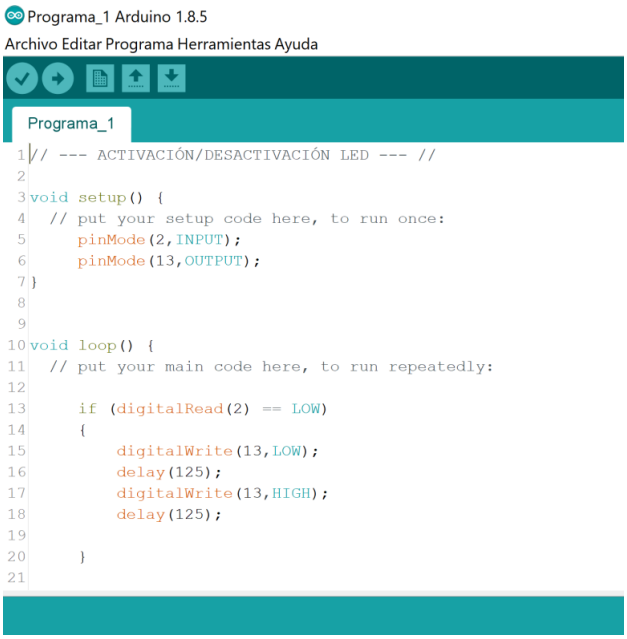
Gebe deinem Haustier Wasser.

2. Aufgabe

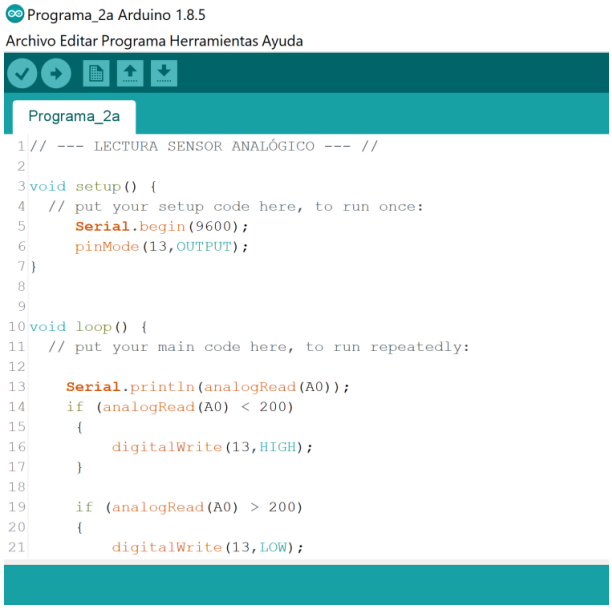
Überlege dir andere Sensoren und Geräte, die zur Simulation anderer Merkmale deines Haustiers verwendet werden können, und bespreche deinen Vorschlag mit der Lehrkraft.

A Programme

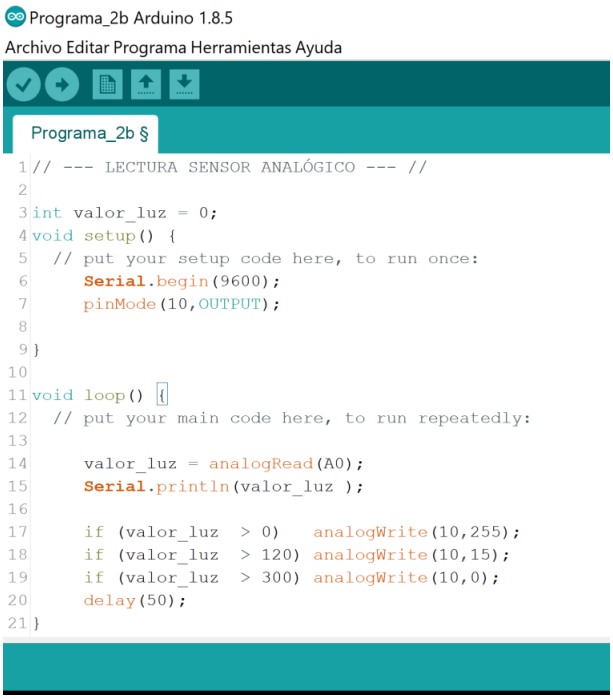
A.1 Programm 1



A.2 Programm 2a



A.3 Programm 2b



A.4 Programm 3a – FICHA 7A (caritas)

```
#include "LedControl.h"
#include "binary.h"

/*
DIN connects to pin 12
CLK connects to pin 11
CS connects to pin 10
*/
LedControl lc=LedControl(12,11,10,1);
// delay time between faces
unsigned long delaytime=1000;
// happy face
byte hf[8]= {B00111100,B01000010,B10100101,B10000001,B10100101,B10011001,
    B01000010,B00111100};
// neutral face
byte nf[8]={B00111100, B01000010,B10100101,B10000001,B10111101,B10000001,
    B01000010,B00111100};
// sad face
byte sf[8]= {B00111100,B01000010,B10100101,B10000001,B10011001,B10100101,
    B01000010,B00111100};
void setup() {
lc.shutdown(0,false);
// Set brightness to a medium value
lc.setIntensity(0,8);
// Clear the display
lc.clearDisplay(0);
}
void drawFaces(){
// Display sad face
```

```
lc.setRow(0,0,sf[0]);
lc.setRow(0,1,sf[1]);
lc.setRow(0,2,sf[2]);
lc.setRow(0,3,sf[3]);
lc.setRow(0,4,sf[4]);
lc.setRow(0,5,sf[5]);
lc.setRow(0,6,sf[6]);
lc.setRow(0,7,sf[7]);
delay(delaytime);
// Display neutral face
lc.setRow(0,0,nf[0]);
lc.setRow(0,1,nf[1]);
lc.setRow(0,2,nf[2]);
lc.setRow(0,3,nf[3]);
lc.setRow(0,4,nf[4]);
lc.setRow(0,5,nf[5]);
lc.setRow(0,6,nf[6]);
lc.setRow(0,7,nf[7]);
delay(delaytime);
// Display happy face
lc.setRow(0,0,hf[0]);
lc.setRow(0,1,hf[1]);
lc.setRow(0,2,hf[2]);
lc.setRow(0,3,hf[3]);
lc.setRow(0,4,hf[4]);
lc.setRow(0,5,hf[5]);
lc.setRow(0,6,hf[6]);

lc.setRow(0,7,hf[7]);
delay(delaytime);
}
void loop(){
drawFaces();
}
```

A.5 Programm 3b – FICHA 7A (corazon)

```
#include "LedControl.h"
LedControl lc=LedControl(12,11,10,1); // Pin and # of Displays
unsigned long delayTime=1000; // Delay between Frames
int heart_stat=5;
int time_stat=0;
// Put values in arrays
byte heart5_icon[] =
{
B01100110,
B11111111,
B11111111,
B11111111,
B01111110,
B00111100,
```

```
B00011000,
B00000000
};
byte heart4_icon[] =
{
B01100110,
B10011001,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart3_icon[] =
{
B01100110,
B10011001,
B10000001,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart2_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart1_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01000010,
B00111100,
B00011000,
B00000000
};
byte heart0_icon[] =
{
B01100110,
```

```
B10011001,
B10000001,
B10000001,
B01000010,
B00100100,
B00011000,
B00000000
};
void setup()
{
lc.shutdown(0,false); // Wake up displays
lc.shutdown(1,false);
lc.setIntensity(0,5); // Set intensity levels
lc.setIntensity(1,5);
lc.clearDisplay(0); // Clear Displays
lc.clearDisplay(1);
}
// Take values in Arrays and Display them
void heart5()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart5_icon[i]);
}
}
void heart4()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart4_icon[i]);
}
}
void heart3()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart3_icon[i]);
}
}
void heart2()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart2_icon[i]);
}
}
void heart1()
{
for (int i = 0; i < 8; i++)
{
```

```
lc.setColumn(0,i,heart1_icon[i]);
}
}
void heart0()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart0_icon[i]);
}
}
void loop() {
// put your main code here, to run repeatedly:
unsigned char ii;
heart5();
delay(1000);
heart4();
delay(1000);
heart3();
delay(1000);
heart2();
delay(1000);
heart1();
delay(1000);
heart0();
delay(1000);
}
```

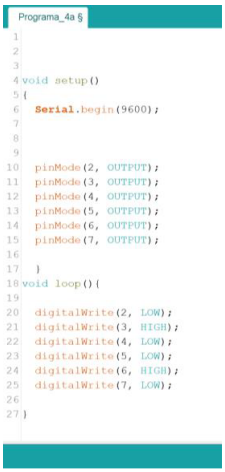
A.6 Programm 3c – FICHA 7A (comida)

```
#include "LedControl.h"
#include "binary.h"
/*
DIN connects to pin 12
CLK connects to pin 11
CS connects to pin 10
*/
LedControl lc=LedControl(12,11,10,1);
// delay time between faces
unsigned long delaytime=1000;
// pollo
byte hf[8]= {B00000000,B00001110,B00011110,B00011110,B0011110,B0011100,↵
    B01100000,B00100000};
// cirera
byte nf[8]={B00000000, B00011000,B00011000,B00100100,B01000010,B11100111,↵
    B11100111,B00000000};
// galeta
byte sf[8]= {B00111100,B01111110,B11111111,B11111111,B11111111,B11111111,↵
    B01111110,B00111100};
// os
byte os[8]= {B00001100,B00001100,B00001111,B00011111,B11111000,B11110000,↵
    B00111000,B00111000};
```

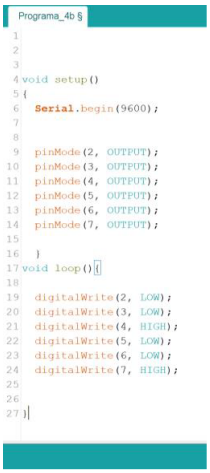
```
void setup() {
lc.shutdown(0,false);
// Set brightness to a medium value
lc.setIntensity(0,8);
// Clear the display
lc.clearDisplay(0);
}
void drawFood(){
// Display galeta
lc.setColumn(0,0,sf[0]);
lc.setColumn(0,1,sf[1]);
lc.setColumn(0,2,sf[2]);
lc.setColumn(0,3,sf[3]);
lc.setColumn(0,4,sf[4]);
lc.setColumn(0,5,sf[5]);
lc.setColumn(0,6,sf[6]);
lc.setColumn(0,7,sf[7]);
delay(delaytime);
// Display cirera
lc.setColumn(0,0,nf[0]);
lc.setColumn(0,1,nf[1]);
lc.setColumn(0,2,nf[2]);
lc.setColumn(0,3,nf[3]);
lc.setColumn(0,4,nf[4]);
lc.setColumn(0,5,nf[5]);
lc.setColumn(0,6,nf[6]);
lc.setColumn(0,7,nf[7]);
delay(delaytime);
// Display pollo
lc.setColumn(0,0,hf[0]);
lc.setColumn(0,1,hf[1]);
lc.setColumn(0,2,hf[2]);
lc.setColumn(0,3,hf[3]);
lc.setColumn(0,4,hf[4]);
lc.setColumn(0,5,hf[5]);
lc.setColumn(0,6,hf[6]);
lc.setColumn(0,7,hf[7]);
delay(delaytime);
// Display os
lc.setColumn(0,0,os[0]);
lc.setColumn(0,1,os[1]);
lc.setColumn(0,2,os[2]);
lc.setColumn(0,3,os[3]);
lc.setColumn(0,4,os[4]);
lc.setColumn(0,5,os[5]);
lc.setColumn(0,6,os[6]);
lc.setColumn(0,7,os[7]);
delay(delaytime);
}
void loop(){
```

```
drawFood();
}
```

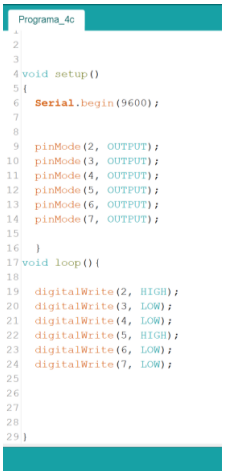
A.7 Programm 4a



A.8 Programm 4b



A.9 Programm 4c - Ficha 7B



A.10 Programm 6

```
#include "LedControl.h"
LedControl lc=LedControl(12,11,10,1); // Pin and # of Displays
unsigned long delayTime=1000; // Delay between Frames
int heart_stat=5;
int time_stat=0;
// Put values in arrays
byte heart5_icon[] =
{
B01100110,
B11111111,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart4_icon[] =
{
B01100110,
B10011001,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart3_icon[] =
{
B01100110,
B10011001,
B10000001,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart2_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01111110,
B00111100,
B00011000,
B00000000
```

```
};
byte heart1_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01000010,
B00111100,
B00011000,
B00000000
};
byte heart0_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01000010,
B00100100,
B00011000,
B00000000
};
void setup()
{
lc.shutdown(0,false); // Wake up displays
lc.shutdown(1,false);
lc.setIntensity(0,5); // Set intensity levels
lc.setIntensity(1,5);
lc.clearDisplay(0); // Clear Displays
lc.clearDisplay(1);
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
pinMode(6, OUTPUT);
pinMode(7, OUTPUT);
}
// Take values in Arrays and Display them
void heart5()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart5_icon[i]);
}
}
void heart4()
{
for (int i = 0; i < 8; i++)
{
```

```
lc.setColumn(0,i,heart4_icon[i]);
}
}
void heart3()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart3_icon[i]);
}
}
void heart2()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart2_icon[i]);
}
}
void heart1()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart1_icon[i]);
}
}
void heart0()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart0_icon[i]);
}
}
void loop() {
// put your main code here, to run repeatedly:
unsigned char ii;
heart5();
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
digitalWrite(6, HIGH);
digitalWrite(7, LOW);
delay(1000);
heart4();
delay(1000);
heart3();
delay(1000);
heart2();
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
```

```
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, HIGH);
delay(1000);
heart1();
delay(1000);
heart0();
digitalWrite(2, HIGH);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
delay(1000);
}
```

A.11 Programm 7

A.12 Programm 8

```
#include "LedControl.h"
LedControl lc=LedControl(12,11,10,2); // Pin and # of Displays
unsigned long delayTime=1500; // Delay between Frames
int heart_stat=5;
int time_stat=0;
// Put values in arrays
byte heart5_icon[] =
{
B01100110,
B11111111,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart4_icon[] =
{
B01100110,
B10011001,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart3_icon[] =
{
B01100110,
B10011001,
```

```
B10000001,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart2_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart1_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01000010,
B00111100,
B00011000,
B00000000
};
byte heart0_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01000010,
B00100100,
B00011000,
B00000000
};
void setup()
{
Serial.begin(9600);
lc.shutdown(0,false); // Wake up displays
lc.setIntensity(0,5); // Set intensity levels
lc.clearDisplay(0); // Clear Displays
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
```

```
pinMode(6, OUTPUT);
pinMode(7, OUTPUT);
}
// Take values in Arrays and Display them
void heart5()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart5_icon[i]);
}
}
void heart4()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart4_icon[i]);
}
}
void heart3()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart3_icon[i]);
}
}
void heart2()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart2_icon[i]);
}
}
void heart1()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart1_icon[i]);
}
}
void heart0()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart0_icon[i]);
}
}
void loop() {
// put your main code here, to run repeatedly:
unsigned char ii;
if (heart_stat == 0)
```

```
{
heart0();
heart_stat= 0;
digitalWrite(2, HIGH);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
}
if (heart_stat == 1)
{
heart1();
heart_stat= 0;
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, HIGH);
}
if (heart_stat == 2)
{
heart2();
heart_stat= 1;
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, HIGH);
}
if (heart_stat == 3)
{
heart3();
heart_stat= 2;
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
digitalWrite(6, HIGH);
digitalWrite(7, LOW);
}
if (heart_stat == 4)
{
heart4();
heart_stat= 3;
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
```

```
digitalWrite(5, LOW);
digitalWrite(6, HIGH);
digitalWrite(7, LOW);
}
if (heart_stat == 5)
{
heart5();
heart_stat= 4;
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
digitalWrite(6, HIGH);
digitalWrite(7, LOW);
}
delay(delayTime);
/* SENSOR LUZ */
if (analogRead(A0) > 200)
{
heart_stat= 5;
}
```

A.13 Programm 8b

```
#include "LedControl.h"
LedControl lc=LedControl(7,6,5,2); // Pin and # of Displays
(::::)
```

A.14 Programm 9 oder program_rfid_read

```
/*
* Initial Author: ryand1011 (https://github.com/ryand1011)
*
* Reads data written by a program such as "rfid_write_personal_data.ino"
*
* See: https://github.com/miguelbalboa/rfid/tree/master/examples/rfi ↵
    d_write_personal_data
*
* Uses MIFARE RFID card using RFID-RC522 reader
* Uses MFRC522 - Library
* ↵
    -----------------------------------------------------------------------------------------

* MFRC522 Arduino Arduino Arduino Arduino Arduino
* Reader/PCD Uno/101 Mega Nano v3 Leonardo/Micro Pro Micro
* Signal Pin Pin Pin Pin Pin Pin
* ↵
    -----------------------------------------------------------------------------------------

* RST/Reset RST 9 5 D9 RESET/ICSP-5 RST
* SPI SS SDA(SS) 10 53 D10 10 10
* SPI MOSI MOSI 11 / ICSP-4 51 D11 ICSP-4 16
```

```
* SPI MISO MISO 12 / ICSP-1 50 D12 ICSP-1 14
* SPI SCK SCK 13 / ICSP-3 52 D13 ICSP-3 15
*/
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9 // Configurable, see typical pin layout above
#define SS_PIN 10 // Configurable, see typical pin layout above
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance
// ↵
    ********************************************************************************

void setup() {
Serial.begin(9600); // Initialize serial communications with the PC
SPI.begin(); // Init SPI bus
mfrc522.PCD_Init(); // Init MFRC522 card
// Serial.println(F("Read personal data on a MIFARE PICC:")); //shows in serial ↵
     that it is ready to read
}
// ↵
    ********************************************************************************

void loop() {
// Prepare key - all keys are set to FFFFFFFFFFFFh at chip delivery from the ↵
    factory.
MFRC522::MIFARE_Key key;
for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
//some variables we need
byte block;
byte len;
int aliment = 11;
MFRC522::StatusCode status;
//-------------------------------------------
// Reset the loop if no new card present on the sensor/reader. This saves the ↵
    entire process when idle.
if ( ! mfrc522.PICC_IsNewCardPresent()) {
return;
}
// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial()) {
return;
}
//----------------------------------------- GET FOOD NAME
byte buffer2[18];
block = 1;
len = 18;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 1, &key, &( ↵
    mfrc522.uid)); //line 834
if (status != MFRC522::STATUS_OK) {
Serial.print(F("Authentication failed: "));
Serial.println(mfrc522.GetStatusCodeName(status));
```

```
return;
}
status = mfrc522.MIFARE_Read(block, buffer2, &len);
if (status != MFRC522::STATUS_OK) {
Serial.print(F("Reading failed: "));
Serial.println(mfrc522.GetStatusCodeName(status));
return;
}
Serial.print(F("Food: "));
// PRINT LAST NAME
for (uint8_t i = 1; i < 16; i++) {
Serial.write(buffer2[i] );
}
aliment = buffer2[0];
if (aliment == '0')
{
Serial.print(F("BONE is GOOD\n"));
}
if (aliment == '1')
{
Serial.print(F("CHERRY is BAD\n"));
}
if (aliment == '2')
{
Serial.print(F("CHICKEN is GOOD\n"));
}
if (aliment == '3')
{
Serial.print(F("COOKIE is GOOD\n"));
}
//----------------------------------------
// Serial.println(F("\n**End Reading**\n"));
delay(1000); //change value if you want to read cards faster
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
}
```

A.15 Programm 10

```
#include "LedControl.h"
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9 // Configurable, see typical pin layout above
#define SS_PIN 10 // Configurable, see typical pin layout above
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance
LedControl lc=LedControl(7,6,5,2); // Pin and # of Displays
unsigned long delayTime=1500; // Delay between Frames
int heart_stat=5;
int time_stat=0;
char menjar[15];
// Put values in arrays
```

```
byte heart5_icon[] =
{
B01100110,
B11111111,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart4_icon[] =
{
B01100110,
B10011001,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart3_icon[] =
{
B01100110,
B10011001,
B10000001,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart2_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01111110,
B00111100,
B00011000,
B00000000
};
byte heart1_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
```

```
B01000010,
B00111100,
B00011000,
B00000000
};
byte heart0_icon[] =
{
B01100110,
B10011001,
B10000001,
B10000001,
B01000010,
B00100100,
B00011000,
B00000000
};
// pollo
byte hf[8]= {B00000000,B00001110,B00011110,B00011110,B0011110,B0011100,
    B01100000,B00100000};
// cirera
byte nf[8]={B00000000, B00011000,B00011000,B00100100,B01000010,B11100111,
    B11100111,B00000000};
// galeta
byte sf[8]= {B00111100,B01111110,B11111111,B01111111,B11111111,B11111111,
    B01111110,B00111100};
// os
byte os[8]= {B00001100,B00001100,B00001111,B00011111,B11111000,B11110000,
    B00111000,B00111000};
void setup()
{
Serial.begin(9600);
SPI.begin(); // Init SPI bus
mfrc522.PCD_Init(); // Init MFRC522 card
// Serial.println(F("Read personal data on a MIFARE PICC:")); //shows in serial
     that it is ready to read
lc.shutdown(0,false); // Wake up displays
lc.setIntensity(0,5); // Set intensity levels
lc.clearDisplay(0); // Clear Displays
}
// Take values in Arrays and Display them
// Display galeta
void cirera()
{
byte nf[8]={B00000000, B00011000,B00011000,B00100100,B01000010,B11100111,
    B11100111,B00000000};
lc.setColumn(0,0,nf[0]);
lc.setColumn(0,1,nf[1]);
lc.setColumn(0,2,nf[2]);
lc.setColumn(0,3,nf[3]);
lc.setColumn(0,4,nf[4]);
```

```
lc.setColumn(0,5,nf[5]);
lc.setColumn(0,6,nf[6]);
lc.setColumn(0,7,nf[7]);
}
void galeta()
{
byte sf[8]= {B00111100,B01111110,B11011011,B11111111,B10101101,B11111111,↵
    B01110110,B00111100};
lc.setColumn(0,0,sf[0]);
lc.setColumn(0,1,sf[1]);
lc.setColumn(0,2,sf[2]);
lc.setColumn(0,3,sf[3]);
lc.setColumn(0,4,sf[4]);
lc.setColumn(0,5,sf[5]);
lc.setColumn(0,6,sf[6]);
lc.setColumn(0,7,sf[7]);
}
void pollo()
{
byte hf[8]= {B00000000,B00001110,B00011110,B00011110,B0011110,B0011100,↵
    B01100000,B00100000};
lc.setColumn(0,0,hf[0]);
lc.setColumn(0,1,hf[1]);
lc.setColumn(0,2,hf[2]);
lc.setColumn(0,3,hf[3]);
lc.setColumn(0,4,hf[4]);
lc.setColumn(0,5,hf[5]);
lc.setColumn(0,6,hf[6]);
lc.setColumn(0,7,hf[7]);
}
void oset()
{
byte os[8]= {B00001100,B00001100,B00001111,B00011111,B11111000,B11110000,↵
    B00111000,B00111000};
lc.setColumn(0,0,os[0]);
lc.setColumn(0,1,os[1]);
lc.setColumn(0,2,os[2]);
lc.setColumn(0,3,os[3]);
lc.setColumn(0,4,os[4]);
lc.setColumn(0,5,os[5]);
lc.setColumn(0,6,os[6]);
lc.setColumn(0,7,os[7]);
}
void heart5()
{
for (int i = 0; i < 8; i++)
{
lc.setColumn(0,i,heart5_icon[i]);
}
}
```

```
154 void heart4()
155 {
156 for (int i = 0; i < 8; i++)
157 {
158 lc.setColumn(0,i,heart4_icon[i]);
159 }
160 }
161 void heart3()
162 {
163 for (int i = 0; i < 8; i++)
164 {
165 lc.setColumn(0,i,heart3_icon[i]);
166 }
167 }
168 void heart2()
169 {
170 for (int i = 0; i < 8; i++)
171 {
172 lc.setColumn(0,i,heart2_icon[i]);
173 }
174 }
175 void heart1()
176 {
177 for (int i = 0; i < 8; i++)
178 {
179 lc.setColumn(0,i,heart1_icon[i]);
180 }
181 }
182 void heart0()
183 {
184 for (int i = 0; i < 8; i++)
185 {
186 lc.setColumn(0,i,heart0_icon[i]);
187 }
188 }
189 void loop() {
190 // put your main code here, to run repeatedly:
191 unsigned char ii;
192 // Prepare key - all keys are set to FFFFFFFFFFFFh at chip delivery from the ↵
    factory.
193 MFRC522::MIFARE_Key key;
194 for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
195 //some variables we need
196 byte block;
197 byte len;
198 int aliment = 11;
199 MFRC522::StatusCode status;
200 //-----------------------------------------
201 // Reset the loop if no new card present on the sensor/reader. This saves the ↵
    entire process when idle.
```

```
202 if ( ! mfrc522.PICC_IsNewCardPresent()) {
203 return;
204 }
205 // Select one of the cards
206 if ( ! mfrc522.PICC_ReadCardSerial()) {
207 return;
208 }
209 //---------------------------------------- GET FOOD NAME
210 byte buffer2[18];
211 block = 1;
212 len = 18;
213 status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 1, &key, &( ↵
    mfrc522.uid)); //line 834
214 if (status != MFRC522::STATUS_OK) {
215 Serial.print(F("Authentication failed: "));
216 Serial.println(mfrc522.GetStatusCodeName(status));
217 return;
218 }
219 status = mfrc522.MIFARE_Read(block, buffer2, &len);
220 if (status != MFRC522::STATUS_OK) {
221 Serial.print(F("Reading failed: "));
222 Serial.println(mfrc522.GetStatusCodeName(status));
223 return;
224 }
225 Serial.print(F("Food: "));
226 // PRINT LAST NAME
227 for (uint8_t i = 1; i < 16; i++) {
228 Serial.write(buffer2[i] );
229 }
230 aliment = buffer2[0];
231 if (aliment == '0')
232 {
233 oset();
234 delay(1000);
235 heart_stat=5;
236 Serial.print(F("BONE is GOOD\n"));
237 }
238 if (aliment == '1')
239 {
240 cirera();
241 delay(1000);
242 heart_stat=0;
243 Serial.print(F("CHERRY is BAD\n"));
244 }
245 if (aliment == '2')
246 {
247 pollo();
248 delay(1000);
249 heart_stat=5;
250 Serial.print(F("CHICKEN is GOOD\n"));
```

```
}
if (aliment == '3')
{
galeta();
delay(1000);
heart_stat=5;
Serial.print(F("COOKIE is GOOD\n"));
}
//---------------------------------------
// Serial.println(F("\n**End Reading**\n"));
delay(1000); //change value if you want to read cards faster
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
//*****************************************************************//
if (heart_stat == 0)
{
heart0();
heart_stat= 0;
}
if (heart_stat == 1)
{
heart1();
heart_stat= 0;
}
if (heart_stat == 2)
{
heart2();
heart_stat= 1;
}
if (heart_stat == 3)
{
heart3();
heart_stat= 2;
}
if (heart_stat == 4)
{
heart4();
heart_stat= 3;
}
if (heart_stat == 5)
{
heart5();
heart_stat= 4;
}
delay(delayTime);
}
```

B Vorlagen

B.1 Vorlage Hund

https://pixabay.com/es/photos/buldog-cachorro-perro-mascota-1047518/

https://pixabay.com/es/photos/perro-animales-continental-bulldog-2437110/

https://pixabay.com/es/photos/cachorro-perro-mascota-animales-1903313/

B.2 Vorlage Katze

https://pixabay.com/es/photos/gato-blanco-animales-mam%C3%ADferos-3591348/

https://pixabay.com/es/photos/gato-animales-caballa-felino-3846780/

https://pixabay.com/es/photos/gato-bebé-dulce-lindo-gatito-4558651/

B.3 Vorlage Schwein

https://pixabay.com/es/photos/lechón-cerdo-joven-el-recién-nacido-3741877/

https://pixabay.com/es/photos/lechón-cerdos-pequeños-mini-lindo-4083870/

https://pixabay.com/es/photos/pincel-oreja-cerdo-4418149/

B.4 Vorlage Papagei

https://pixabay.com/es/photos/loro-arco-iris-loros-australia-686100/

https://pixabay.com/es/photos/perico-periquito-amarillo-447710/

https://pixabay.com/es/photos/animales-ara-macao-pico-ave-84485/

B.5 Vorlage Maus



https://pixabay.com/es/photos/animales-criatura-bicho-domesticado-1239398/

https://pixabay.com/es/photos/animales-atractivo-hermosa-brown-1238239/

https://pixabay.com/es/photos-hámster-roedor-animal-pet-roedores-1596819/