

**Teil 4**

# Lektion

# 7

## Uhr-Karten-Maschine

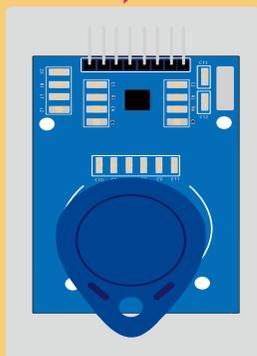
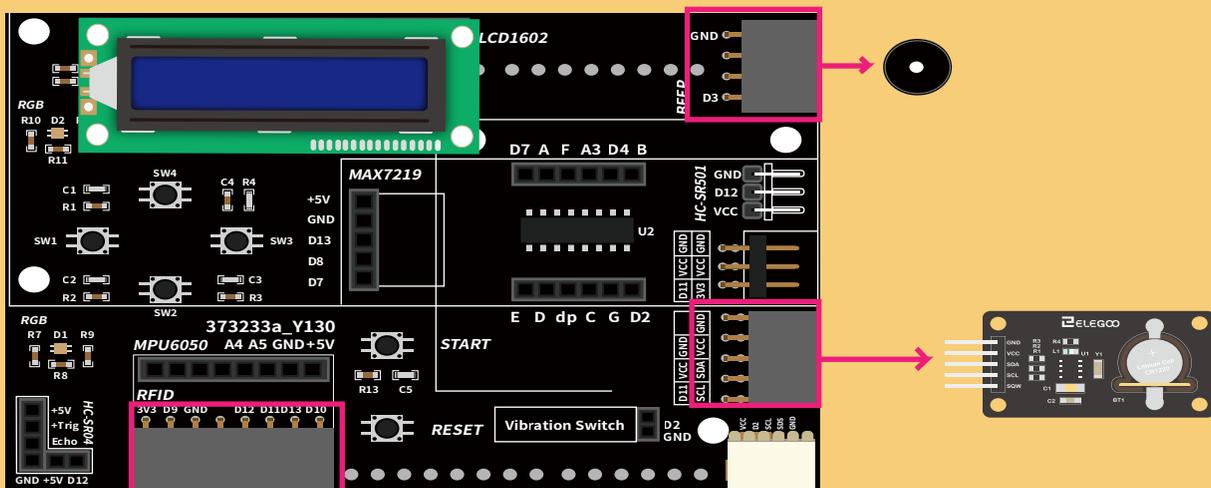
## Zusammenfassung:

Durch das Studium des Sketches der Uhr-Karten-Maschine werden wir unser Verständnis für die praktische Anwendung des LCD1602-Anzeigemoduls, des RC522-RF-Kartenmoduls und des Summers vertiefen.

## benötigte Komponenten:

- (1) x Arduino Uno
- (1) x RFID
- (1) x Passive buzzer
- (1) x ALL IN ONE Sensor Shield
- (1) x LCD1602
- (1) x Clock Module

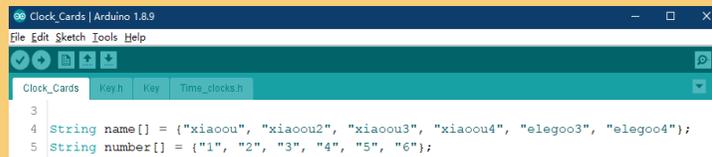
## Schaltplan:



Tips: Bitte fügen Sie das erweiterte Board in die UNO ein.

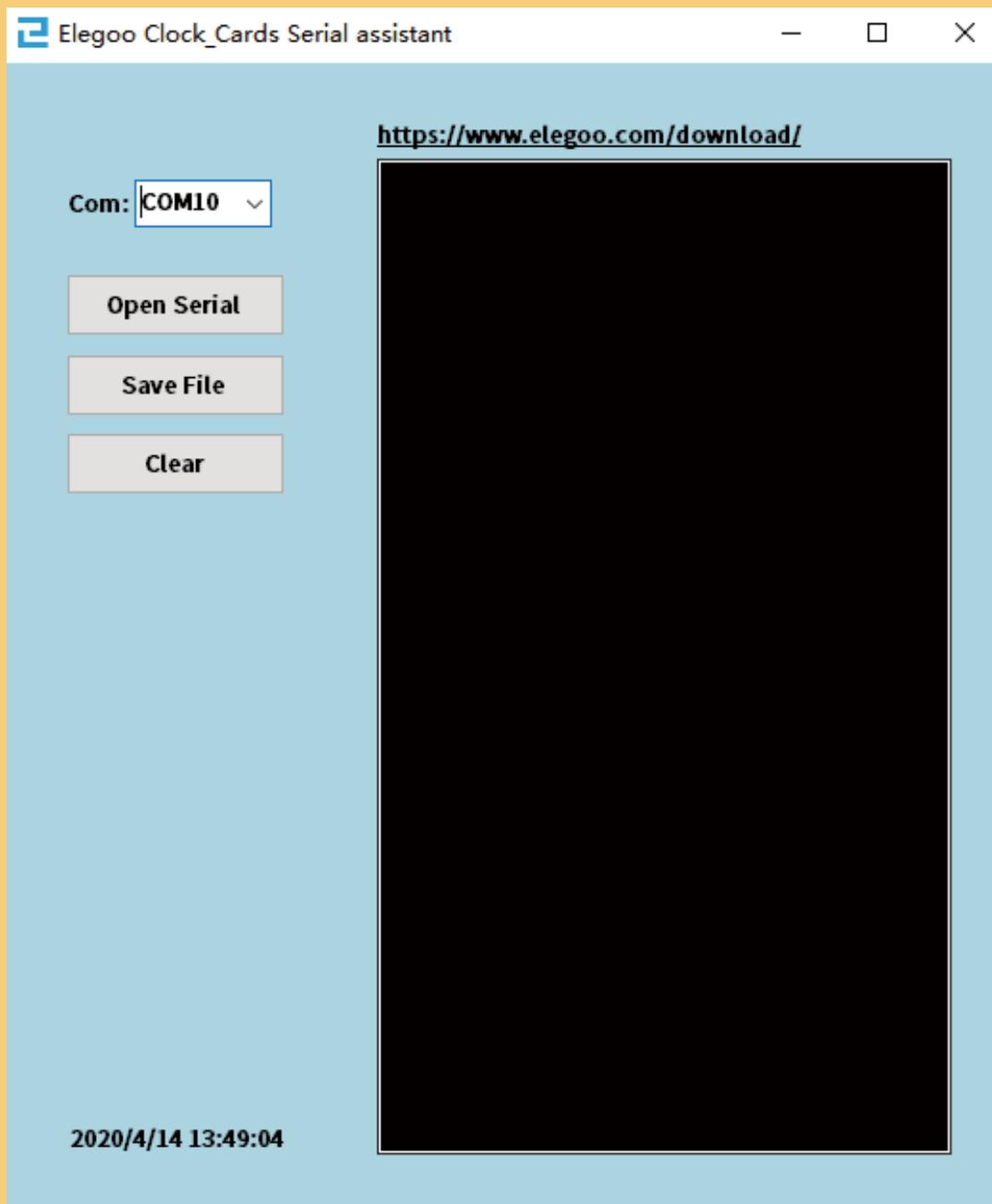
## Realisierung der Uhren-Karten-Maschine:

Geben Sie zuerst den Namen und die Mitarbeiteridentifikationsnummer des Mitarbeiters ein und laden Sie dann den Sketch herunter.

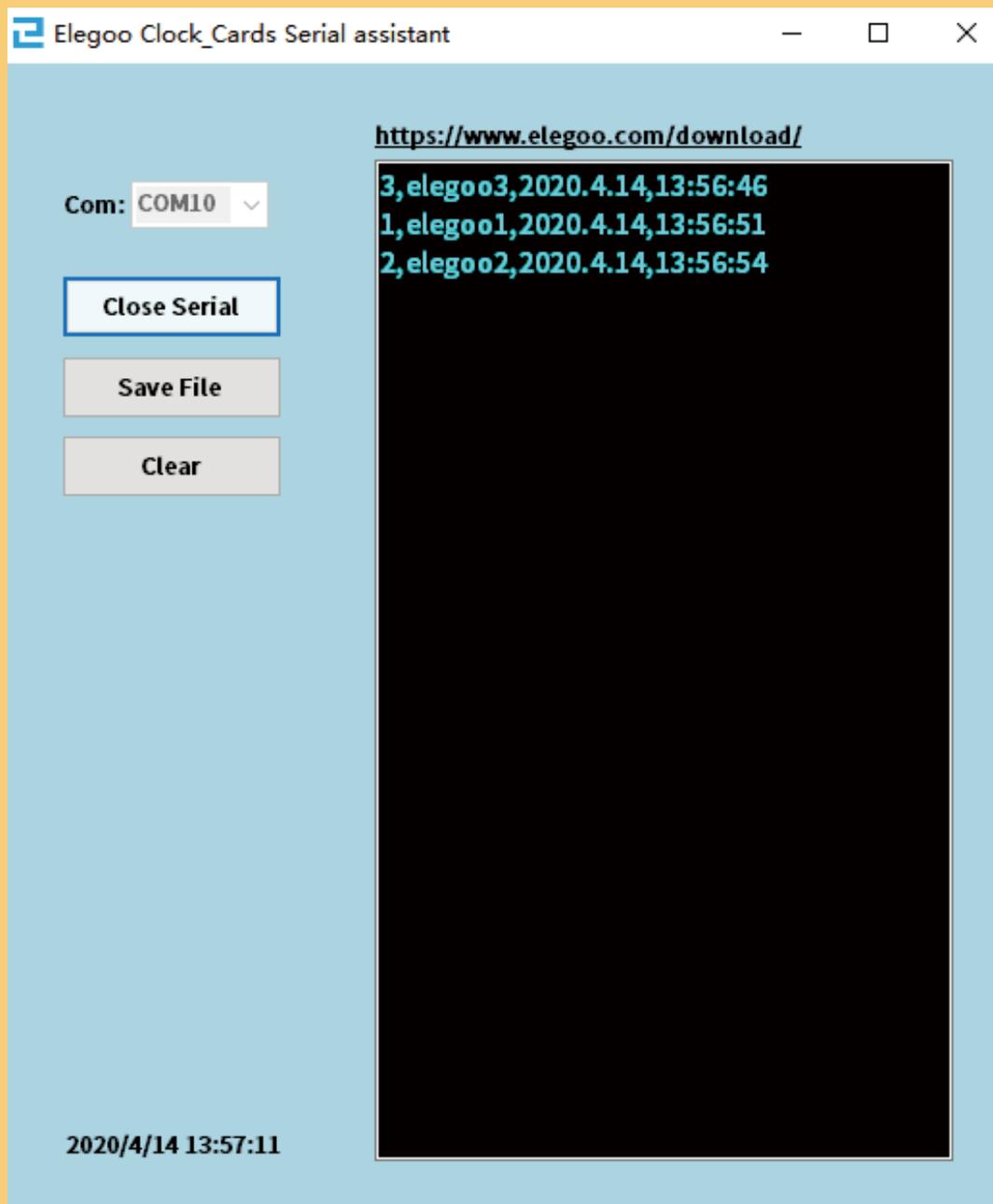


```
Clock_Cards | Arduino 1.8.9
File Edit Sketch Tools Help
Clock_Cards Key.h Key Time_clocks.h
3
4 String name[] = {"xiaou", "xiaou2", "xiaou3", "xiaou4", "elegoo3", "elegoo4"};
5 String number[] = {"1", "2", "3", "4", "5", "6"};
```

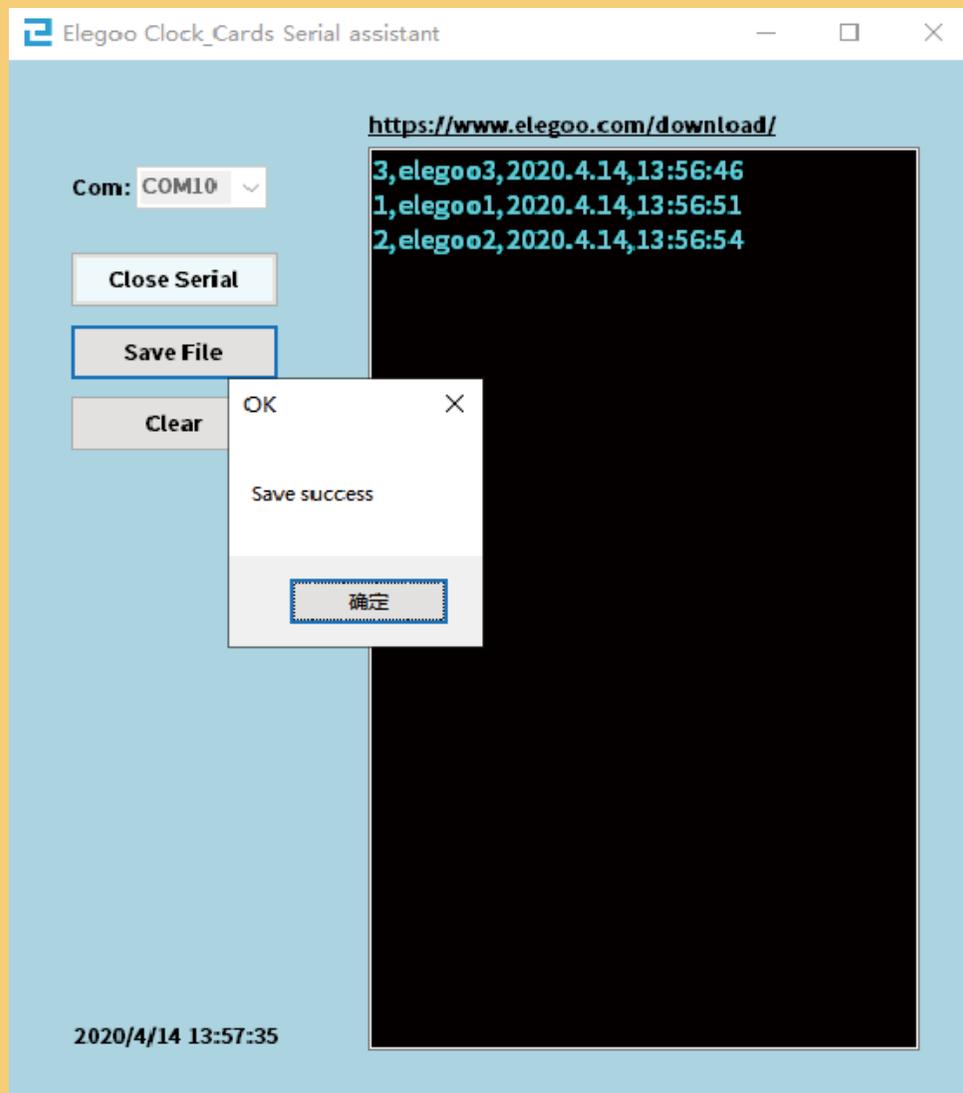
Trennen Sie nach dem Herunterladen des Sketches nicht das USB-Kabel: öffnen Sie die unterstützende Software und drücken Sie auf "Open Serial".



Wechseln Sie in den Eingabemodus und geben Sie die Informationen nacheinander ein. Warten Sie bei der Eingabe etwa zwei Sekunden und die Eingabeaufforderung auf dem Display zeigt an, dass die Eingabe erfolgreich war. Wenn die Eingabe abgeschlossen ist, drücken Sie die Taste erneut, um den Eingabemodus zu verlassen. Wischen Sie zu diesem Zeitpunkt mit der aufgezeichneten RF-Karte und Sie können den Namen, die Mitarbeiteridentifikationsnummer und die Zeit des Wischkontaktes der Person abrufen, welche die Karte in der Software durchgezogen hat.



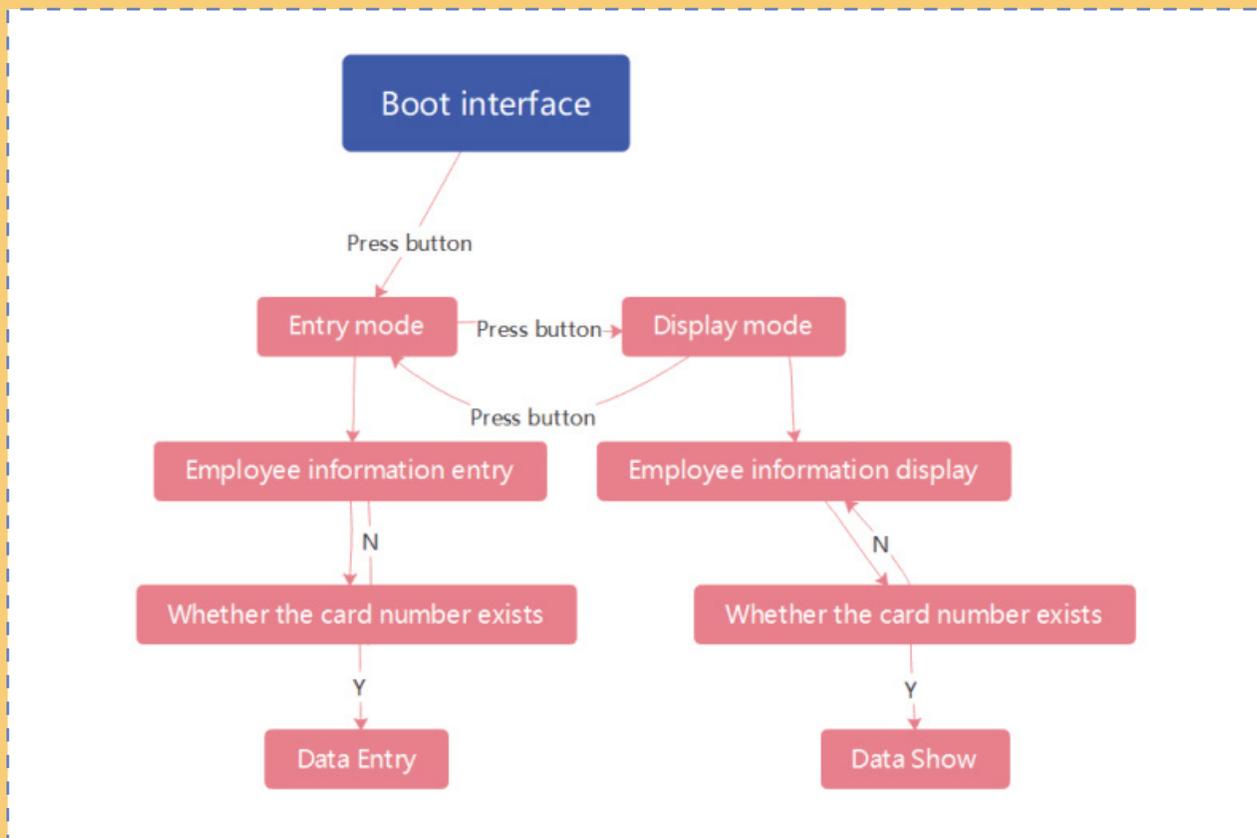
Wenn Sie schließlich auf "Save File" klicken, werden die angezeigten Informationen im CSV-Format gespeichert.



DailyData.csv	2020/4/9 15:07
MySerial.exe	2020/4/14 13:54

	A	B	C	D	E
1	Number	Name	Date	Time	
2		3 elegoo3	2020. 4. 14	13:56:46	
3		1 elegoo1	2020. 4. 14	13:56:51	
4		2 elegoo2	2020. 4. 14	13:56:54	
5					

## Implementierungs-Schemata:



### Prozess-Ablauf:

- 1. Fügen Sie zunächst die zu verwendenden Bibliotheksdateien und die Objekte hinzu, welche zum Aufrufen der entsprechenden Bibliotheksfunktionen definiert werden müssen.

```
//In Time_clocks.h
```

```
#include <Wire.h>
#include <DS3231.h>
#include <LiquidCrystal.h>
#include <SPI.h>
#include <MFRC522.h>
#include "pitches.h"
```

```
#define RST_PIN  A1
#define SS_PIN  10
```

```
void LCD_ShowDate(RTCDateTime dt);
```

```
DS3231 clock;
RTCDateTime dt;
LiquidCrystal lcd(A3, 2, 4, 7, 8, A2);
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

```
//In Clock_Cards

#include "Time_clocks.h"
#include "PinChangeInt.h"

void setup() {
  lcd.begin(16, 2);
  Serial.begin(9600);
  clock.begin();
  clock.setDateTime(__DATE__, __TIME__);
  SPI.begin(); // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card
  pinMode(A0, INPUT_PULLUP);
  attachPinChangeInterrupt(A0, button_choose_interrupt, FALLING );
}
```

## ■ 2. Interface-Start

```
//In Clock_Cards

void loop() {
  Start_interface();
}

void Start_interface()
{
  if (flag_start == 1)
  {
    lcd.setCursor(3, 0);
    lcd.print("Starting up.");
    delay(2000);
    flag_start = 0;
    lcd.clear();
  }
}
```

- 3. Definitionen und Initialisierungen mit Löschen des Bildschirms bei jeder Änderung eines Ereignisses.

```
//In Key.h

#ifndef KEY_H
#define KEY_H

int time_button_choose=0;
int last_time_button_choose=0;

#define ELIMINAT_THE_JITTER_OF_BUTTON_CHOOSE
(200 < (abs(time_button_choose - last_time_button_choose)))

#endif
```

```
//In Key

#include "Key.h"

void button_choose_interrupt ()
{
    time_button_choose = millis();

    if(ELIMINAT_THE_JITTER_OF_BUTTON_CHOOSE)
    {
        last_time_button_choose = time_button_choose;
        if(mode==DISPLAY_MODE)
        {
            mode=INPUT_MODE;
            lcd.clear();
        }
        else if(mode==INPUT_MODE)
        {
            mode=DISPLAY_MODE;
            lcd.clear();
        }
    }
}
```

- 4. Schalten Sie den Modus gemäß dem Schlüsselstatus um.

```
//In Clock_Cards

void Mode_Chose(int mode)
{
    switch (mode)
    {
        case DISPLAY_MODE:
            .....
        case INPUT_MODE:
            .....
        default: break;
    }
}
```

- 5. Erstellen Sie ein Array zum Speichern von Mitarbeiterinformationen.

```
//In Clock_Cards

String name[]={"xiaoou","xiaoou2","xiaoou3","xiaoou4","elegoo3","elegoo4"};
String number[]={"1","2","3","4","5","6"};
```

- 6. Im Eingabemodus wird eine verknüpfte Liste der entsprechenden Größen entsprechend der Arraygröße für die Informationsspeicherung erstellt.

```
//In Clock_Cards

void Mode_Chose(int mode)
{
    if (cnt < (sizeof(name) / sizeof(String)))
    {
        Data_Input(name[cnt], number[cnt]);
    }
    delay(100); break;
}
```

- 7. Stellen Sie fest, ob die Kartennummer vorhanden ist, erstellen Sie eine verknüpfte Liste, um die Informationen zu speichern, falls die Karte vorhanden ist. Andernfalls überspringen Sie, wenn die Karte nicht gültig ist.

```
//In Clock_Cards

void Data_Input(String name, String number)
```

## Die Vor- und Nachteile von Array und verknüpfter Liste:

### Array-Definition:

Vorteile: Einfach zu bedienen, Abfrageeffizienz ist höher als verknüpfte Liste, Speicher als kontinuierlicher Bereich. Nachteile: Feste Größe, nicht für dynamische Speicherung geeignet, nicht für dynamische Addition geeignet.

### verknüpfte Liste:

Advantages: Dynamic addition and deletion, variable size.

Disadvantages: It can only be accessed through sequential pointers, and the query efficiency is low.

**Linked list: In essence, it is a node to establish a connection with a pointer.**

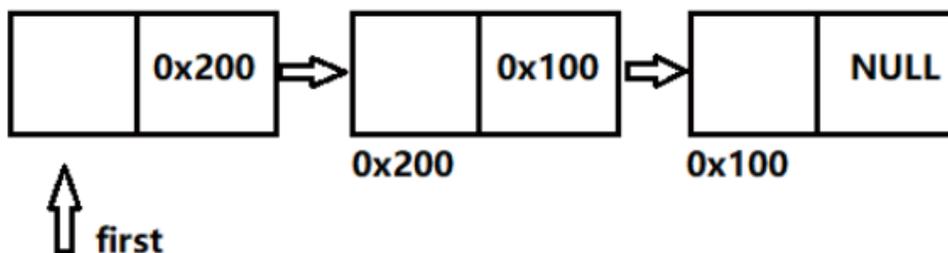
```
struct node
```

```
{
```

```
    ElemType data; //Data field
```

```
    struct node *next; //Pointer field
```

```
};
```



**The pointer next stores the address of the next node.**

- 8. Programmiermodus anzeigen: Durchlaufen Sie die verknüpfte Liste, um die Daten gemäß der Kartennummer zu lesen.

```
//In Clock_Cards
```

```
void Data_Display()
```