Teil 4

Lektion 4

Bohnen erhalten

Überblick:

Durch das praktische Erlernen des Spielprojekts von “Bohnen erhalten” können wir das Verständnis für die praktische Anwendung des Max7219-Moduls, des MPU6050-Moduls und des Summers verbessern. Zusätzlich folgen wir der Programmieridee mit Verständnis und lernen so den Arduino besser kennen.

Benötigte Komponenten:

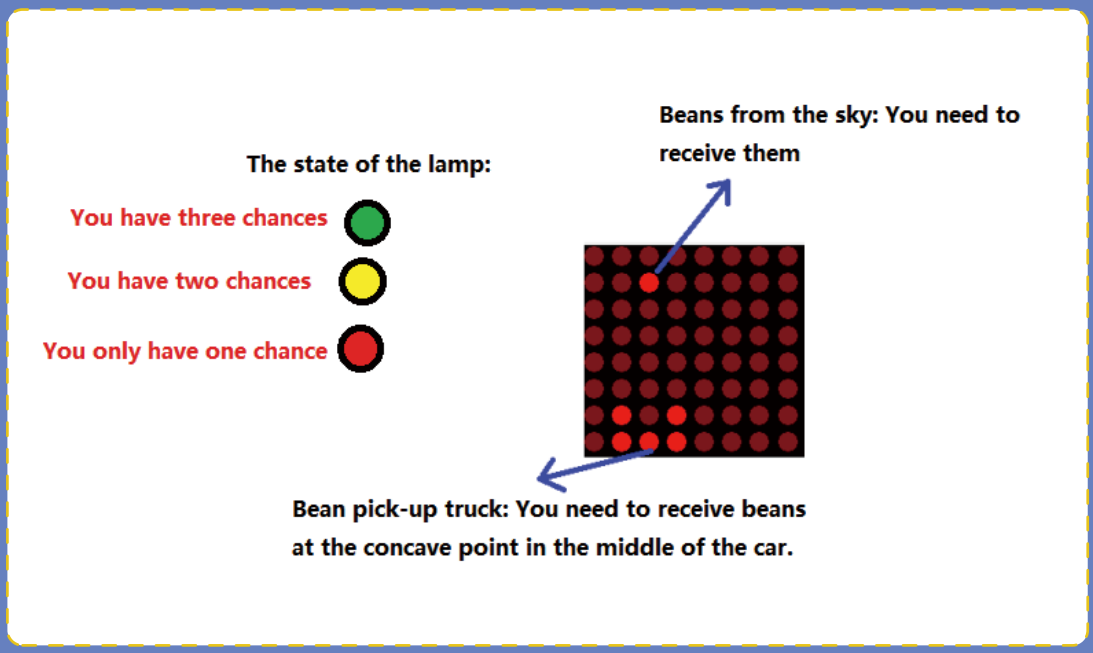
(1) X Elegoo UNO R3

(1) X MPU 6050 module

(1)X MAX7219 module

(1) X RGB LED

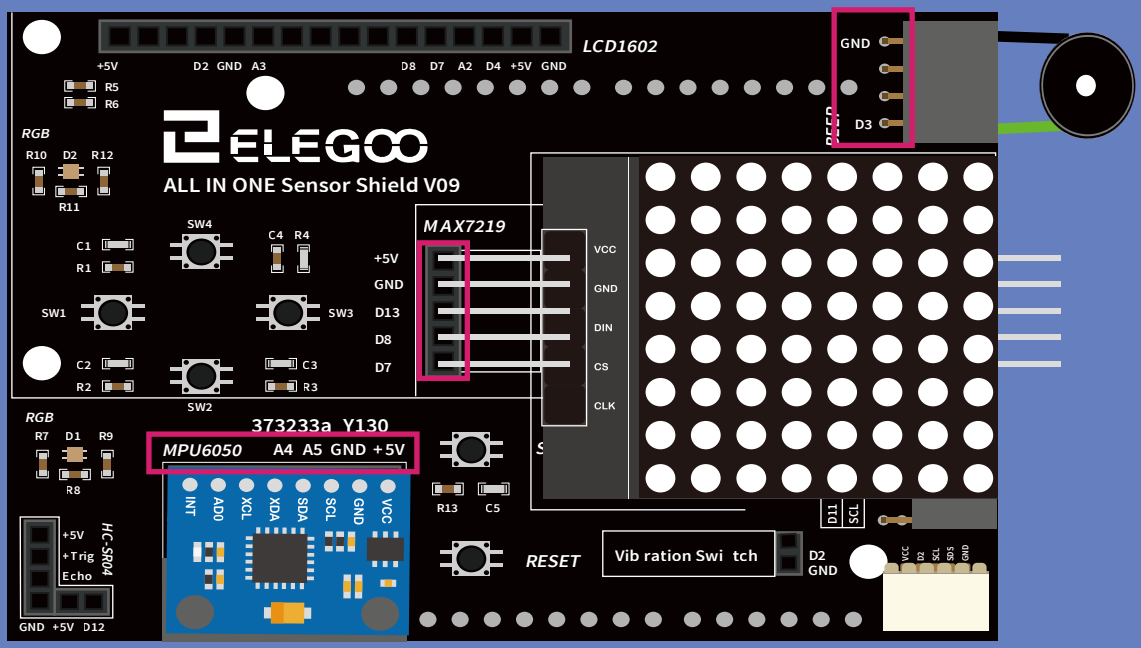
(1) X Passive Buzzer



Schütteln Sie das Board nach links und rechts, um die Bewegung des Fahrzeugs zu steuern. Die Bohnen werden gefangen, wenn die mittlere Konkave mit den Bohnen ausgerichtet ist. Wenn es Ihnen gelingt, mehr als 10 Bohnen zu fangen, wird ein lächelndes Gesichtsmuster angezeigt, andernfalls wird das Spiel beendet. Wenn das Spiel vorbei ist, schütteln Sie die Erweiterung nach oben und unten, um das Spiel neu zu starten.

Schaltplan:

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



Stellen Sie sicher, dass diese Bibliotheken hinzugefügt sind, bevor Sie das Programm

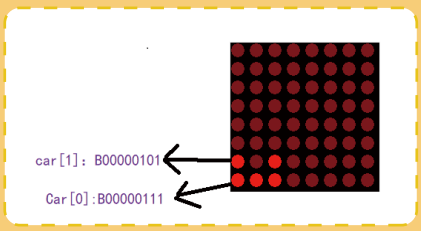
- Part1 (demo1):
  Bewegungskontrolle des Autos
  /**/Please open demo1 which in the code

- Zuerst lassen wir ein Auto auf dem MAX7219 anzeigen: also müssen wir ein Array von Autos definieren.

- Int car [2]={B00000111, B00000101}.
  Weil der Index "[num]" von Null an zählt.
  Car [0] is "B00000111"
  Car [1] is "B00000101"
  There are two. So num is 2.
  So far, the car have been made

```
void game_init() //car init
{
 lc.setRow(0,7,car[0]);
 lc.setRow(0,6,car[1]);
}
```

Theory (PWM)



- Als nächstes programmieren wir die Bewegung des Autos nach links und rechts.Details warden in diesen Funktionen definiert:
  Control_right()
  Control_left()

- Um eine Bewegung zu erreichen, benutzen wir "<<" um B00000111 nach links zu bewegen.
  Beispiel: B00000111<<1 becomes B00001110


B00000111


B00001110

```
  void Control_right()
{
  ......
     else{
      car[0]=car[0]<<1;
      car[1]=car[1]<<1;
     }
  ......
}
```

- Wenn es also auf dem Bildschirm angezeigt wird, bewegt es sich ein Bit.
- Wenn wir in der Lage sind, das Auto so zu steuern, dass es sich nach links und rechts bewegt, müssen wir weitere Aspekte berücksichtigen.
- Wir müssen jedes Mal, wenn sich das Auto bewegt, einen Kantenkollisionstest durchführen. Andernfalls kann es sich an den Rand links oder rechts bewegen oder auch ganz verschwinden.
- Wenn sich das Fahrzeug in die folgende Position bewegt, sollten wir diese Bewegung abbrechen und die Position des Fahrzeugs zurücksetzen.

```
  void Control_right()
{
  if(AcY>5000)
 {
    if(car[0]==B00000011)
   {
    car[0]=B00000111;
    car[1]=B00000101;
   }
    if(car[0]==B11000000)
     {
      car[0]=B00000011;
      car[1]=B00000010;
     }
  ......
 }
}
```

- The first part of the detailed code, please open Demo1 in code.

Part2 (demo2):

Bohnen fallen

- /**/Please open demo2 which in the code
- Weitere Informationen finden Sie in unserer Funktion pro_random_beans().

```
void random_init()
{randomSeed(analogRead(0));} //random init
```

- 1.1.Da das Erscheinen von Bohnen zufällig ist, müssen wir randomSeed(analogRead(0)) hinzufügen, um eine Zufallszahl zu erzeugen, und diese als Spalte auswählen, die Bohnen generiert.
- 2.Stellen Sie dann den Bereich der Zufallszahlen ein.

```
uint8_t bean_x = 0;    //Storage location: At which point16_t the bean falls on the x-axis
uint8_t bean_y = 0;    //Storage location: At which point16_t the bean falls on the y-axis
void beans_init()
{bean_y = random(7);
 bean_x=0;}
```

- Weil unser Bildschirm acht Spalten hat (0 ~ 7)

- 3.Wenn die Bohnen erzeugt werden und klar ist, in welcher Spalte sie produziert werden, fallen sie allmählich vom Himmel.
- Wir realisieren das wie folgt: Die einzelnen Licht-Punkte in derselben Spalte werden nacheinander ein- und ausgeschaltet.
- Hierzu verwenden hier die for-Schleife.
  Im Übrigen wird eine Variable bean_speed als Intervall zwischen dem Ein- und Ausschalten der Lichtpunkte in derselben Spalte festgelegt, das meint die Geschwindigkeit, mit der die Bohnen fallen.

```
void pro_random_beans()//produce ramdon beans
{
  beans_init();
 for(bean_x=0;bean_x<=7;)
 {
   beans_falling();
  }
}
void beans_falling()
{
 lc.setLed(0,bean_x,7-bean_y,true);
 delay(bean_flick_speed);
 lc.setLed(0,bean_x,7-bean_y,false);
 delay(bean_flick_speed);
 delay(300);
 bean_x++;
}
```

- Im zweiten Teil des detaillierten Codes öffnen Sie bitte demo2 im Code.

Part 3(demo 3):

Receive bean

- /**/Please open demo 3 which in the code
- Zu diesem Zeitpunkt fügen wir die in Teil1 geschriebene Funktion control_right(), control_left() und game_init() zur Funktion pro_random_beans() von Teil 2 hinzu, um den Effekt des Aufnehmens von Bohnen zu erzeugen.
- Im dritten Teil des detaillierten Codes öffnen Sie bitte demo3 im Code.

```
void pro_random_beans()//produce ramdon beans
{
  beans_init();
 for(bean_x=0;bean_x<=7;)
 {
   update_Ac();
   Control_right();
   Control_left();
 game_init();
   beans_falling();
   }
}
```

Part 4(demo 4):

Animation and Sound produzieren

- /**/Please open demo 4 which in the code
  / **/ We can change the following variable to modify the animation and sound speed.

```
int16_t er_show_speed=150;  //enlarge_reduce_display speed
int16_t heart_show_speed=150;
int16_t game_show_speed=80;
int16_t music_speed=100;
int16_t car_display_speed=85;
```

- Dieser Teil des Codes realisiert hauptsächlich den Soundeffekt, den Lichteffekt und die Bildschirmanzeige beim Empfangen von Bohnen.
- Im vierten Teil des detaillierten Codes öffnen Sie bitte Demo 4 im Code.

Part 5 (receive_beans_G):

receive_beans_G

- /**/Please open receive_beans_G in the code folder
- Der letzte Teil ist eine Kombination der vorherigen Teile. In diesem Teil lernen wir hauptsächlich, wie man den Mechanismus des Empfangens von Bohnen erkennt.
- Prinzip: Wir beurteilen, ob sich das Auto[0] an derselben Stelle befindet (x []), wenn die Bohnen in die letzte Reihe fallen.

```
//// When the beans fall to the last grid, the car must be in the X [] position to receive the beans correctly.
uint8_t x[]={B00000011,B00000111,B00001110,B00011100,B00111000,B01110000,B11100000,B11000000};
```

- If car [0]==x [bean_f] picks up beans successfully
  If car [0]! = x [bean_f] failed to pick up beans
- Im fünften Teil des detaillierten Codes öffnen Sie bitte receive_beans_G im Code.

```
void case_ending_judge()
{
  if(bean_x>7)//When the beans fall to the last grid
 {
   switch(bean_y)//Determine which column is the last grid
   {
     case x1:if(car[0]!=x[bean_y]) {case_run();}
           else {success_receive_voice();
               beans_received_num++; } break;
     ……
    }
   }

 }
```