4 Gewinnt mit dem AZ-Touch Mod

Wie Sie ggf. schon mitbekommen haben, bin ich nicht nur Blogger, sondern auch Familienvater. Damit muss man sein Hobby und das Familienleben irgendwie miteinander kombinieren und kleinere Spielereien für die Kinder programmieren. In diesem Fall das beliebte Spiel „4 Gewinnt“, welches es schon in diversen Formen und Ausführungen gibt. In diesem Fall bringen wir das Spiel auf den AZ-Touch Mod mit 2.4“-Display oder 2.8“-Display und dabei sollen Sie gleichzeitig noch etwas über mehrdimensionale Arrays und rekursive Funktionen lernen

Die Hard- und Software für diesen Blog

Damit Sie diesen Blog nachbauen können, brauchen Sie die Komponenten aus Tabelle 1:

Pos	Anzahl	Bauteil
1	1	ESP32 NodeMCU Module WLAN WiFi Development Board mit CP2102 (Nachfolgermodell zum ESP8266)
2 oder	1	AZ-Touch MOD Wandgehäuseset mit 2,4 Zoll Touchscreen für ESP8266 und ESP 32
3	1	AZ-Touch MOD Wandgehäuseset mit 2,8 Zoll Touchscreen für ESP8266 und ESP32
4 (optional)	1	12V Netzteil zur Spannungsversorgung

Tabelle 1: Bauteile für 4 Gewinnt

An Software brauchen Sie:

- Arduino IDE (https://www.arduino.cc/en/Main/Software), hier am besten die aktuellste Version herunterladen
- Die Bibliothek TFT_eSPI mit allen Abhängigkeiten. Wie das geht, zeigen wir u.a. hier.

Die Grundlagen

Das mehrdimensionale Array

Wenn Sie nicht gerade aus der IT kommen, aber Programmieren eines ihrer Hobbies ist, werden Sie wahrscheinlich erst einmal bei dem Begriff „mehrdimensionales Array“ stutzen. Bei einem meiner letzten Spielblogbeiträge TicTacToe, haben ich ein normales Array vom Datentyp Integer genutzt, siehe Code 1.

```
int iSetMark[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};
```

Code 1: Array vom Datentyp Integer

Hierbei handelt es sich um ein sogenanntes eindimensionales Array, wobei nur eine Datenreihe existiert. In Zeile 29 bei dem Spiel „4 Gewinnt“ wird ein zweidimensionales Array angelegt, siehe Code 2.

```
byte bMatrix[NUMROW][NUMCOLUMN];
```
Code 2: Zweidimensionales Array

Vom Prinzip können Sie sich ein zweidimensionales Array wie eine Excelliste vorstellen. Sie tragen in jede Spalte und Reihe einen Wert ein, der dann wiederrum ausgelesen oder beschrieben werden kann. Im Grunde kann das so weit getrieben werden, dass sie ein n-dimensionales Array anlegen können, siehe Code 3.

```
Datatype Array[Erste Ebene][Zweite Ebene][Dritte Ebene]….[n-1 Ebene][n-te Ebene];
```
Code 3: n-dimensionales Array

Kommentiert [AW1]: ich glaube, n-1te Ebene ist nicht korrekt, es müsste glaub ich nur n-te Ebene sein. Sonst ist es doppelt gemoppelt

Kommentiert [JAW2R1]: In der Mathematik und Informatik nutzt du normalerweise genau diese Schreibweise um klarzumachen, dass du die vorletzte und letzte Datenreihe aufzeigst.

Das Auslesen und Schreiben solcher Arrays kann recht schwierig werden, je nachdem wie komplex das Array ist. Zum Resetten des zweidimensionalen Arrays von „4 Gewinnt“ reichen zwei ineinander verschachtelte for-Schleifen, siehe Code 4.

```
/*
* ================================================================
* Function:     resetMatrix
* Returns:      void
* Description:  Reset the matrix
* ================================================================
*/
void resetMatrix()
{
  Serial.println("----- Reset Matrix -----");
  for(int iColumn = 0; iColumn < int(NUMCOLUMN); iColumn++)
    for(int iRow = 0; iRow < int(NUMROW); iRow++)
      bMatrix[iRow][iColumn] = 0;
  showMatrix();
  Serial.println("-------------------------");
}
```
Code 4: Das zweidimensionale Array von "4 Gewinnt" resetten

Um die verschachtelte for-Schleife noch besser zu verstehen, können wir sie aufrollen. Im Kern finden wir den Befehl: bMatrix[iRow][iColumn] = 0;
Die Speicherplätze eines Arrays werden in den eckigen Klammern angegeben. Je mehr eckige Klammern, desto mehr Ebenen (wie oben beschrieben). Zwei Ebenen stellen wie gesagt eine Tabelle dar. Man surft dann mit den Werten in den Klammern durch die Zeilen und Spalten beginnend bei 0:
bMatrix[0][0] = 0;
bMatrix[0][1] = 0;
…
bMatrix[1][0] = 0;
bMatrix[1][1] = 0;
…
usw.
Eine der beiden for-Schleifen kann nun dafür genutzt werden, durch eine Zeile zu laufen (iterieren). Die andere for-Schleife zählt dann durch die Spalten. In diesem Fall ist die innere Schleife für die Zeilen zuständig, die äußere für die Spalten.
Ein dreidimensionales Array könnten Sie sich nun vorstellen wie eine zweite Tabelle, die hinter der ersten liegt. Im Kopf der for-Schleife wird der Startwert, der Endwert und dann die Schrittweite angegeben.
Um alle Werte in einem Array mit 0 zu definieren, sollte man in C bzw. C++ diesen Weg gehen.

Schauen Sie sich einfach mal die Funktion **checkForWinner()** an, bei der das zweidimensionale Array ausgelesen wird, um einen möglichen Gewinner zu ermitteln.

Der rekursive Funktionsaufruf

Wenn schon die Funktion **checkForWinner()** angesprochen wurde, soll auch gleich das Thema rekursiver Funktionsaufruf näher erklärt werden. In vielen unserer Blogbeiträge werden sie etwas finden wie in Code 5.

```
bool checkVertical(int iRow, int iColumn)
```

Code 5: Beispiel einer Funktion

Zunächst wird definiert, welcher Rückgabewert die Funktion hat, im oberen Beispiel ein bool. ~~(also True oder False, bzw. HIGH oder LOW, bzw. 0 oder 1).~~ Direkt danach folgt der Name der Funktion, über welche die Funktion im Quellcode aufgerufen wird. Danach folgt in den Klammern, welche Daten als Kopie, als Referenz oder Pointer genutzt werden. Damit haben Sie erst einmal das Prinzip einer „normalen“ Funktion geklärt. Ein rekursiver Funktionsaufruf heißt so, weil die Funktion sich selbst wieder aufruft. Das können Sie am besten in Code 6 sehen.

Kommentiert [JAW3]: Ein Bool ist normalerweise immer True oder False, alles andere ist ein Cast durch die jeweilige Entwicklerumgebung, die das intern regelt.

```
/*
* ================================================================
* Function:       checkVertical
* Returns:        true if there is a winner else false
* INPUT iRow:     Current row
* INPUT iColumn:  Current column
* REF iSum:       Sum of current equal positions
* Description:    Recursive function to check vertical win
* ================================================================
*/
bool checkVertical(int iRow, int iColumn, int &iSum)
{
  if(bMatrix[iRow][iColumn] != bPlayerMove || bMatrix[iRow][iColumn] == 0)
    return false;
  else
  {
    iSum++;
    if(iSum == 4)
      return true;
    else
      return checkVertical(iRow, iColumn+1, iSum);
  }
}
```

Code 6: Beispiel einer rekursiven Funktion

Das gefährliche an dieser Art von Aufruf ist, dass Fehler nur schwer zu finden sind und Sie ggf. eine ungewollte Endlosschleife programmieren. Daher muss die Abbruchbedingung sauber programmiert sein, siehe die innere if-Abfrage aus Code 6, welche ein true zurückliefert und den rekursiven Aufruf beendet. An dieser Stelle wird die Referenz iSum geprüft, ob iSum den Wert 4 erreicht hat.
Seien Sie also vorsichtig, wenn Sie rekursiven Funktionen programmieren.

Das Spiel auf den MicroController übertragen

Damit Sie „4 Gewinnt“ spielen können, laden Sie bitte das zip-Paket von meiner GitHub-Seite herunter. Entpacken Sie diese zip-Datei und öffnen Sie den Ordner direkt danach. Sie werden dort zwei Dateien vorfinden:

- 4Gewinnt.ino: Das Programm für die Arduino IDE
- User_Setup.h: Die Konfigurationsdatei für das Display

Die User_setup.h ersetzen Sie bitte im Order C:\User**IHR-Name**\Documents\Arduino\libraries\TFT_eSPI, siehe Abbildung 1. Es macht Sinn, zuvor die dort befindliche Datei mit gleichem Namen umzubenennen, z,B. in **User_Setup_old.h**.

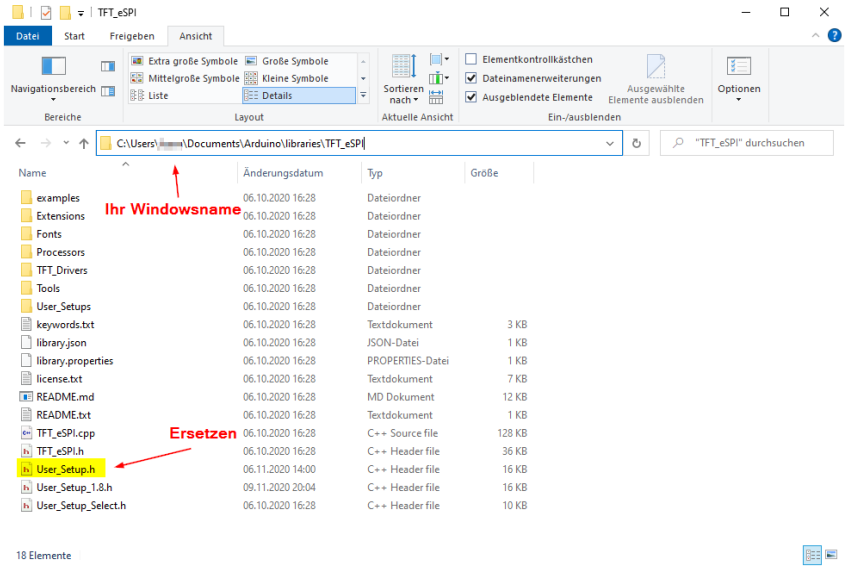

Abbildung 1: User_Setup.h überschreiben

Danach öffnen Sie den Quellcode in der Arduino IDE und wählen den passenden Micro Controller nach eBook-Anleitung aus. Übertragen Sie das Programm auf den Micro Controller und nach einem Neustart sollten Sie die Aufforderung zur Kalibrierung sehen, siehe Abbildung 2.

Kommentiert [AW4]: welches E-Book?

Abbildung 2: Kalibrierungsaufforderung auf dem Display

Sollte beim Kompilieren ein Fehler auftreten, z.B. dass TFT_eSPI die Funktion **calibrateTouch()** nicht kennt, dann haben Sie die User_Setup.h nicht korrekt ersetzt, oder die User_Setup.h befindet sich im selben Verzeichnis wie die *.ino-Datei, was nicht sein darf.

Ist dieser Schritt abgeschlossen, achten Sie auf den grünen Pfeil bei den roten Kästchen in der Ecke, dann wird Ihnen direkt danach der Startscreen gezeigt, siehe Abbildung 3.

Abbildung 3: Startscreen von „4 Gewinnt“

Durch das Drücken vom Button „Start“ wird direkt das Spielfeld aufgebaut, siehe Abbildung 4.

Abbildung 4: Das Spielfeld

Oben sehen Sie Zielscheiben, um Ihren Spielstein zu setzen, rechts sehen Sie, welcher Spieler gerade an der Reihe ist. Sie und ihr Spielgegner sehen also zu jeder Zeit, wer am Zug ist und natürlich auch die gelegten Steine, siehe

Abbildung 5~~Abbildung 5~~.

Abbildung 5: Fortgeschrittenes Spiel mit allen nötigen Informationen

Machen Sie sich bei dem Spiel keine Gedanken das richtige Kästchen zu erwischen, je nachdem welche Spalte Sie oder ihr Spielgegner auswählen, wird automatisch die richtige Position vom Spielstein ermittelt. Wenn eine Spalte voll ist, ist ein weiterer Spielzug in dieser Spalte auch nicht mehr möglich, wie beim echten Spiel.

Hat einer der beiden Spieler gewonnen oder ist das Spielfeld gefüllt und keiner konnte gewinnen, wird ein entsprechender Endscreen dargestellt, siehe Abbildung 6

Abbildung 6: Spiel zu Ende, Spieler 2 hat gewonnen

Wir wünschen viel Spaß beim Spielen und eine kleine wichtige Information noch zum Schluss:
Da das 2.4“ und das 2.8“-Display dieselbe Auflösung haben muss nicht im Quellcode umständlich etwas verändert werden. Durch die Kalibrierung am Anfang, können Sie entweder das Az-Touch mit 2.4“- oder 2.8“-Display einsetzen.

Weitere Projekte für AZ-Delivery von mir, finden Sie unter https://github.com/M3taKn1ght/Blog-Repo.