Sachthemen für die Doku: automatisches Gewächshaus

Programmieren:

- Tasterpin als Eingang, durch internem Pullup auf high gezogen, damit ein definierter Zustand herrscht

- Zustandsautomat für die Pumpenfunktion, weil spart Rechenzeit, definierte Vorgehensweise – erst Messen, wenn nass wieder messen, wenn trocken nächster Zustand pumpen usw

- nutzen des Timers über die Funktion millis(), man braucht kein delay und der µC kann weiterarbeiten, wenn die Zeiten nicht abgelaufen sind, ACHUNG! Millis() läuft nach 50 Tagen über und geht auf Null

- erstellen von Funktionen für bessere Lesbarkeit des Programms und wiederkehrende Aufrufe, spart auch Schreibarbeit

- Programm modularisieren → Funktionen auslagern

- die „echte“ main liegt unter $HOME/arduino/hardware/arduino/avr/cores/arduino/main.cpp ; man hat im Sketch nur Zugriff auf die Funktionen setup und loop, daher ist auch das normale Vorgehen zum Auslagern von Fuktionen über Definition in einer xyz.c und Deklaration in einer xyz.h und das alles in der main.c zu includieren nicht möglich

- Zum Auslagern von Funktionen [zB. xyz()] haben wir ein Header-Datei erstellt, wo die Funktionsdefinitionen drinstehen und in dem Hauptsketch über #include „xyz.h“ eingebunden werden.

Quellen:
https://www.mikrocontroller.net/articles/Umstieg_von_Arduino_auf_AVR
https://www.arduino.cc/reference/de/language/functions/time/millis/

Elektronik:

- EMV-Problem bei den Wasserpumpen, beim Einschalten der Pumpen über ein Relais, schaltete sich der Arduino aus. Bei Fehlersuche ergab sich, immer wenn das Kabel in der Nähe des µC lag, strürzte er ab, lag es weit weg, dann funktionierte es gut. Nach Messungen mit dem Oszi, konnte man die HF-Störung sehen und diese mit einer Entstör-Drossel (Spule im Hin- und Rückzweig plus Keramikkondensator parallel) soweit reduzieren, das sie keinen Einfluß auf die Arbeit des Arduino mehr haben.
Quelle: „EMV-Störungen/Prozessankopplung“ Skript von Kurt Jankowski-Tepe

- Da die Heizmatte 4,5A zieht, brauchen wir einen extra Stecker und extra Netzteil, die SubD-Stecker dürfen max mit 1A belastet werden

- 12V Netzteil versorgt alles was 12V braucht (Lüfter,…) direkt und geht in einen DC-DC Spannungswandler, der aus 12V → 5V macht und alle 5V-Teile versorgt (Pumpe, Sensoren, Display…) und auch den Arduino selbst, dann der 5V Ausgang des Arduino, kann auch als 5V

Eingang verwendet werden, um den Arduino mit 5V zu versorgen, dann ist man unabhängig vom USB und EXT-Eingang (7V-12V) → entnommen aus der Arduino-Refferenz

- Das Display wird über i2c angesteuert um Leitungen zu sparen. Dazu wird ein I2C-Adapter verwendet, welcher aus den 8 Datenleitungen, nur zwei Leitungen macht. Unsere Module verwenden einen IC „NXD PCF 8574T“. Das ist ein Portexpander von parallel zu I2C.

- für den Helligkeitssensor wird der IC LM393 mit einem Fotowiderstand genutzt. Hell sind 0,3V und Dunkel sind 5V

- Warum Arduino?
Kennenlernen der Hardware und die Arbeitweise des Arduino; leichte Einarbeitung; gute Referenz auf Deutsch und Englisch in Netz; vorgegebene Funktionen dadurch kein Eingreifen in Register zB bei PWM, Interrups…; IDE in Jave läuft auf Windows, Mac und Linux; µC vom Atmel leicht zu Programmieren über USB-Anschluß – kein Programmer notwendig (Bootloader); es gibt viele Shields Sensoren Aktoren fertig im Netz zB von Adafruit zu kaufen;

Quelle: „Mikrocontroller vderstehen und Anwenden“ von Clemes Valens

- Steuerelektronik wird in einem seperaten Kasten eingebaut, um es vor Wasser/Feuchte zu schützen

-