Direct Mode Servo

In der Servo Platine 510 wird das digitale Signal vom ARDUINO erst vo einem WS2811 Chip in drei analoge Werte umgewandelt (R,G,B). Diese drei anaolgen Signale werden dann von einem Attiny-85 eingelesen und in digitale Signale für die drei angeschlossenen Servos umgewandelt. Dieser etwas umständliche Weg mußte gewählt werden, da die direkte Verarbeitung des ARDUINO Signals im Attiny nicht möglich erschien, da der Attiny dafür zu langsam war.

Mit dem Direct Mode Servo hat es Eckhard geschafft, mit dem Attiny die ARDUINO Signale direkt auszuwerten und daraus die digitalen Servo Signale zu erzeugen. Dieser direkte Weg hat mehrere Vorteile:

1. Einsparung des WS2811 - kleinere Platine
2. genauere Reaktion der Servos, da die Umwandlung in Analogsignale und Verarbeitunmg im Attiny zu Ungenauigkeiten führt.
3. einfachere Programmierung der Endlagen

Für den Aufbau einer Direct-Mode-Servo-Platine gibt es zwei Möglichkeiten:

1. die neue Platine 511 (in Entwicklung - noch nicht verfügbar)
2. Anpassung der Platine 510

Platine 511

Beschreibung folgt, wenn die Platine verfügbar ist

Umbau PLatine 510 zu Direct Mode Servo

Da sich die Verfügbarkeit der echten 511 Direkt-Mode-Servo Platine etwas hinzieht (Schuld liegt bei mir, bzw. meinem Zeit-Budget für unser Hobby ;)) bin ich schon gefragt worden, wie die Anders-Bestückung der bisherigen 510er Servos Platine, quasi als Übergangs-Lösung, genau aussieht. Daher gibt es hier jetzt die 510 Umbau-Beschreibung, die ich auch Harold, als er mit dem Python Programm Generator, für den Direkt Mode-Servo begonnen hat, gegeben habe:

Basis ist eine modifizierte MobaLedLib 510 Servoplatine. Bei dieser muss der WS2811 Chip weggelassen, bzw. entfernt werden. Ausserdem müssen die Widerstände R5/R9/R10 weggelassen, bzw. entfernt werden. R5 und R9 würden die Lichtbus DI/DO Leitungen unzulässig dämpfen. Statt R5 zu entfernen reicht es auch, die Brücke SERVO1 wieder zu öffnen. Statt R10 mit 1K sollte ein 10K Widerstand eingebaut werden, wenn wir den ATTiny85 Pin 1 nicht zum I/O umkonfigurieren (dieser Pin wird nicht gebraucht), sondern als RESET belassen.

Außerdem müssen statt des WS2811 Chips zwei Brücken eingebaut werden. Und zwar am Bauelementeplatz des WS2811 (egal ob DIP-8, oder SOP-8 unten) von Pin 1 zu Pin 6 und von Pin 2 zu Pin 5. Bitte unbedingt die Zählweise der Pins beachten, die Brücken dürfen sich NICHT kreuzen!

Bei einer Neubestückung können auch der Widerstand R1 und der Kondensator C1 weggelassen werden; bei einer Abänderung der alten Variante stören sie aber nicht!

Dazu kommen Bilder, die den Umbau, bis auf die 4,7K Servo-Pull-Up Widerstände R6, R7, R8, zeigen:

Eine Diskussion zu dem Umbau findet Ihr hier:
https://www.stummiforum.de/t226083f195-pyMLL-fuer-Windows-LINUX-und-MAC.html

Dieses Bild zeigt schematisch, was zu ändern ist:

Programmieren kann man den ATTiny85, mit dem bekannten 400er Progger, aus dem Python Programm Generator heraus; fast an der selben Stelle, wie auch Hardies Variante. Es heißt dort „Servo 2“

Es gibt aber auch eine einfachere Alternative zu dem 400er Progger, die mit dem Direct-Mode-Servo funktioniert: Einfacher-Attiny-Programmer

Zudem WICHTIG: Die Chip Ein- und Ausgänge des ATTiny85 haben NICHT die selben physikalischen Eigenschaften der DI und DO Leitungen von echten WS2811/12 LEDs! Die Tiny Pins sind eigentlich für die Kommunikation von Chips untereinander, auf einer gemeinsamen Platine, gedacht. Man kann sie nicht, wie die WS2811/12 LEDs bis zu 2m voneinander entfernen! Ich habe aber erfolgreich Reichweiten mit bis zu 50-70cm getestet.

Ergänzung: Eigentlich wisst ihr alle, die ihr eine normale MLL Hauptplatine verwendet, doch gut bescheid! Denn wenn ihr die Heartbeat LED dort weglasst (oder ohne MLL Hauptplatine mal eben einen freien Arduino dafür verwendet), hat die Reichweite zur ersten Nutz-LED andere Grenzen, als mit Heartbeat LED! Der Nano hat hier nämlich die selbe Physik, wie der DM Tiny. Nur macht der DM Tiny es nicht nur einmal vorwärts, sondern auch noch einmal rückwärts!

Man sollte also, bei der DM modifizierten 510 Platine und später auch der orginären 511 Platine, nicht zu weit entfernt davor und dahinter im Strang echte WS2811/12 LEDs haben!

Folgende Jumper müssen geschlossen werden: SERVO, SERVO2, SERVO3 und nach Bedarf (letzter 510(1) in der Kette) TERM.

SERVO1 ist egal, wenn R5 nicht bestückt ist und muss geöffnet werden, wenn R5 vorher mal bestückt wurde.

Adressierung der Servos

Bei der modifizierten Platine belegt jeder Servo eine RGB-Adresse. (Bei der originalen Platine hatte das gesamte Modul eine RGB-LED Adresse und die Servos wurde durch die R-G-B Kanäle angesprochen). Die Adressen bei der modifizierten Platine sind fortlaufend. D.h. erste Servo-Adresse 5, dann haben die beiden anderen Servos die Adresse 6 und 7. Diese Adresse ist auch bei der Endlageneinstellung und in der Servo-Animation angegeben werden, damit der richtige Servo angesprochen wird.

Einstellung der Endlagen

Die Einstellung der Endlagen erfolgt über die „Servo2“ Seite des pyProgrammGenerators. Die Beschreibung findest Du hier: pyProgrammGenerator - Servo2 Seite

Programmierung von Animationen

Die Programmierung von Animationen erfolgt durch den Servo-Animations Macro im pyProgrammGenerator. Detail ssind hier zu finden: pyProgrammGenerator - Servo Animation

From:
https://wiki.mobaledlib.de/ - **MobaLedLib Wiki**

Permanent link:
https://wiki.mobaledlib.de/anleitungen/spezial/pyprogramgenerator/direct-mode-servo

Last update: **2024/07/01 11:30**