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.

Einstellung der Endlagen

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

In pyMobaLedLib die Seite „Servo2“ öffnen.

Es ist aber ganz einfach: - Die Servo Position in die Mitte stellen. - Die Servo Control Betriebsart auf:

„Training End Pos einstellen“ - Von der Mitte ausgehend langsam zur gewünschten Endposition fahren. Der Servo sollte dieser Bewegung folgen. - Wenn die erste Endposition erreicht ist, „Enter drücken“ - zur zweiten Endposition fahren. - Nochmal „Enter“ drücken - Betriebsart wieder auf „Normal“ stellen.

Achtung: Der nachfolgende Text ist eine Sammlung von Hinweisen aus der Diskussion. Der Text wird noch etwas besser strukturiert…

Genau genommen ist der Bereich von 0 bis 255, beim Direct Mode Servo, eine Art „Zahlenwerte Matroschka“! (die russische „Puppe in der Puppe“)

Wenn du den Servo im Normalbereich mit den Endlagen trainierst, dann entspricht der Bereich von 0 bis 255 den PWM Werten von 1ms bis 2ms. Wenn du das selbe mit der Einstellung (spezial) tust, dann umfassen die Werte 0 bis 255 die PWM Werte von 0,5ms bis 2,5ms.

Doch egal, welche Millisekunden Werte du für die Endlagen festlegst, der gewählte Bereich hat vom Python Programm Generator aus wieder den Bereich von 0 bis 255. Das wäre auch dann der Fall, wenn der Bereich zwischen den gewählten Endlagen sehr schmal ist. Damit erhält man in dem schmalen Bereich eine sehr hohe Genauigkeit! Umstellen auf Endlagen Schieberegler auf die eine Endlage einstellen, Enter drücken, Schieberegler auf die zweite Endlage stellen und wieder Enter drücken - fertig. Ist das so richtig?

Dirket nach dem Flashen des Attiny85 und im Normalmodus bewegt sich der Servo mit dem Schieberegler von 0 bis 255 nur etwa 30°. Durch drücken von Enter geht er nochmal etwas weiter so dass ich in etwa auf die eigentlichen Maxima komme. Passt das soweit?

Ja, ob die Programmierung erfolgreich war, kannst du ja daran sehen, dass sich nun der Normalbereich 0.255 innerhalb der eingestellten Endlagen bewegt. Danach sollte der DM Servo auch im Betriebsmode ansprechbar sein. Die 30 Grad default sind eine Sicherheitsfunktion, damit man sich nicht gleich seine Mechanik beschädigt.

Ein DM Servo belegt den Adressraum einer RGB LED mit drei Mal 8 Bit bzw. Werten von 0..255! Im ersten Kanal kommt eine Steuerinformation und eine CRC Prüfsumme, im zweiten Kanal ein Stellwert von 0.255 und der dritte Kanal ist bei der Std. MLL derzeit nicht genutzt. Außerdem hat die Std. MLL die ersten beiden Kanäle technisch vertauscht. Deine Beobachtung könnte also richtig sein, wenn sich „Helligkeit“ auf alle drei Kanäle gleichzeitig einer RGB LED bezieht.

so, die Kanäle sind geklärt. Es ändert sich an einer RGB-LED an einem WS2811-Modul der rote Kanal und grün wird langsam heller. Mit einer WS2812 sind rot und grün vertauscht, also alles so wie es soll. Das passt erstmal mit deiner Beschreibung.

Der zweite und dritte Servo haben sich auch geklärt. Ich hatte die 510 vorher an das Ende meiner Kette gehängt, wodurch sie die Adresse 47 bekommen hat. Adresse 48 und 49 sind dann Servo 2 und 3. Aber bei Adresse 48 wird zumindest bei mir die erste LED auf Kanal 1 angesprochen. Ich habe die 510 nun als erste an Kanal 0 gehängt und nun werden nach der Heartbeat auf 1, 2 und 3 die drei Servos angesprochen.

Aber es ist bei allen drei Servos so, dass die Einstellung bei Normal Position erstmal nur in sehr begrenztem Bereich funktioniert und das auch nur in dem ich zu 0 gehe und dann mehrmals zwischen 1 und 0 wechsle, damit der Servo immer weiter zum Anschlag läuft. Das Gleiche am anderen Ende bei 255. Und ich bekomme mit meinen SG90 Servos nur 90° maximal hin. Das sollten eigenlich ca. 170° sein.

Programmierung der Endlagen: mache ich das richtig: Training End Pos angeben, erst auf 0 stellen, Enter drücken, 255 einstellen und wieder Enter drücken? Nach dem ich das getan habe bewegen sich die Servos unter Normal Position wieder nur wie vorher beschrieben.

ProgrammGenerator: Da ich die 510 jetzt nach zwei WS2812 und der Heartbeat auf der Hauptplatine angeschlossen habe, habe ich mit der V5.3.0J die Tabelle auf den ESP32 hochgeladen und siehe da, die Servos funktionieren und zwar laufen sie im vollen 90° wie unter Servo Test 2 eingestellt. Also hat die Programmierung der Endlagen irgendwie funktioniert. Wenngleich nur 90° statt 170°.

Du bist, für den derzeitigen Programm-Stand, viel zu schnell mit der Maus unterwegs! Hintergrund: Modellbau Servos benötigen ein ständiges Refreshen des PWM Signals. Bei der Nutzung für die Modellbahn hat sich aber eingebürgert, das PWM Signal irgendwann abzuschalten, damit die Servos nicht am mechanischen Anschlag brummen. FastLED/MLL sendet das WS2811 Signal auch in einer refresh Loop und der pyPG synthetisiert ein Pattern, mit dem der Stellwert übertragen und eine bestimmbare Zeit nach der letzten Stellbewegung wieder abgeschaltet wird. (üblich ist, bei der Modellbahn, ca. 1 Sek.)

Aber: das Trainieren der Endlagen ist eigentlich der Farbtest und hier werden NUR EINZELNE Werte in den WS2811 Strang übertragen, wenn der pyPG vom User eine Aktion bekommen hat. „old school“ Servos speichern aber nicht den letzten empfangenen PWM Wert und laufen von alleine weiter, bis der erreicht ist, sondern vergleichen immer nur, im Moment des PWM Empfangs, die Pulsweite mit der aktuellen Stellung und treiben dann den Motor an.

Du musst also (derzeit) den pyPG Regler so langsam bewegen, dass das Servo auch hinterher kommt! Nach meinen Messungen und meiner Erfahrung haben SG90 Servos, bei der Servo Norm-PWM von 1-2ms, nur 90 Grad Stellwinkel! Die 170 Grad Aussage kommt dadurch zustande, dass fast alle eben nicht 1-2ms PWMs machen, sondern mehr! Das gibt es beim DM Servo auch und heißt „Training Pos Mode (spezial)“ Warnung: (gerade auch sehr kraftvolle) Servos können sich, wenn sie an den mechanischen Anschlag gefahren werden, selbst beschädigen!

ch habe die 510 an einer Verteilerplatine mit separater 5V-Versorgung. Das sollte kein Problem sein. Aber ich hatte für die Tests drei Servos parallel am Laufen, vielleicht ist dann der Strom doch mal zu gering. Werden die Factory Defaults dann auch gleich wieder ins EEPROM geschrieben? Sonst müssten beim nächsten Aufruf die Endlagen wieder erneut ausgelesen

Tatsächlich haben die Tinys hinter einer echten physikalischen WS2811 oder WS2812B LED eine ganz passable Reichweite! Die nächste WS2811, hinter dem Tiny, kann auch ganz ordentlich weg sein (z.B. 70cm kein Problem), aber ATMega238 auf ATTiny85 ist irgendwie seltsam. Es wundert mich ein wenig, denn ATTiny auf ATTiny, also zwei meiner Direct Mode Servo Platinen, vertragen durchaus einen halben Meter zwischeneinander! wenn das EEPROM als inkonsistent erkannt wird, erhält das EEPROM die selben default Werte, wie bei der ersten Inbetriebnahme. Die alten Endlagen sind also weg!

Normalerweise befindet sich der ARDUINO im sogenannten „Effekt-Modus“. In diesem Modus spielt er die Effekte z.B. Heartbeat, Servo Animation usw ab. Er reagiert in diesem Modus auch auf Stellbefehle, die über die USB-Schnittstelle kommen.

Damit ich die angeschlossenen Objekte, RGB-Leds, WS2811-Platinen oder Servoplatinen direkt ansteuern kann, kann ich den ARDUINO in den sohenanten „Direkt-Modus“ umschalten. In diesem Modus werden keine Effekte mehr abgearbeitet, sondern ich kann direkt jede LED adressieren und die Farbe, bzw. beim Servo die Position einstellen. Du kannst den Modus unten in der Status-Leiste links

sehen. Dort steht z.B. „Status: Verbunden Comx - Direct Mode“ oder „Status: Verbunden Comx - Effect Mode“

Auf der Servo2 Seite muß ich den Servo direkt anteuern. Also schalte ich dort auf den „Direct Mode“ um und ide Heartbeat LED geht aus.

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?rev=1719763893

Last update: **2024/06/30 16:11**