Sterowniki robotów
Projekt Openservo

Jarosław Toliński

31 maja 2012

Spis treści

1 Wstęp teoretyczny

1.1 Czym jest Openservo?

OpenServo jest otwartym projektem zakładającym projektowanie, budowanie oraz programowanie układów pełniących role sterowników do serwomechanizmów, prowadzonym przez Barry Carter' a,

1.2 Założenia projektowe

Celem projektu było stworzenie sterownika seromechanizmu zdolnego do komunikacji po RS485. Na podstawie celów projektu wyznaczono następujące założenia projektowe:

- zaprojektowanie układu elektronicznego oraz płytki PCB na podstawie projektu OpenServo,
- zmaksymalizowanie wykorzytania dostępnego w ramach projektu OpenServo kodu i dodanie do niego założonych wczesniej funkcjonalności.

1.3 Różnice

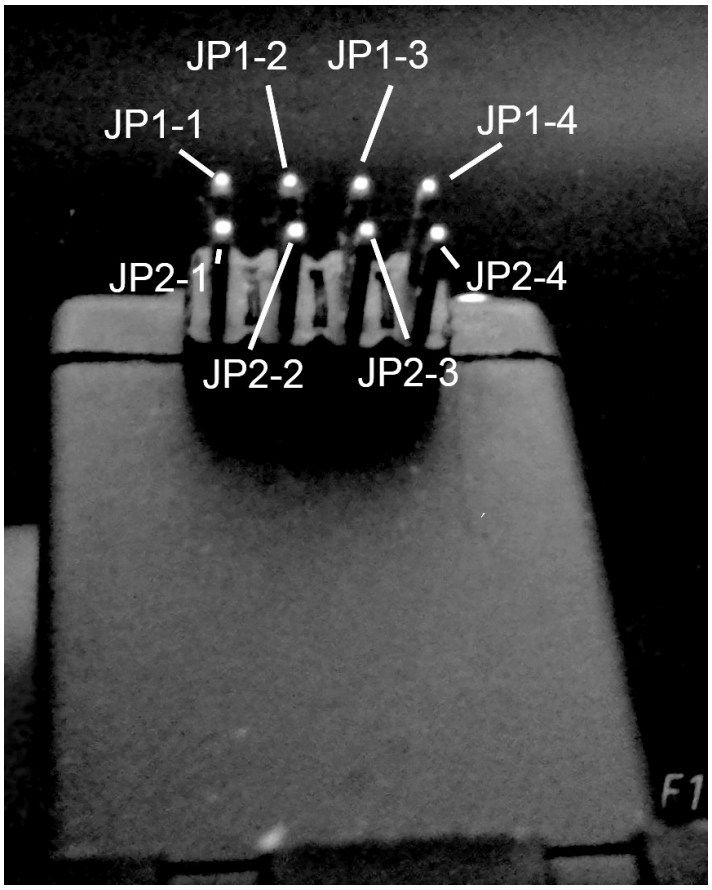
Aby zapewnić możliwość komunikacji tworzonego OpenServa z użyciem RS485, do oryginalnego projektu elektroniki oraz firmware-u OpenServa wprowadzono kilka zmian:

- dodano układ MAX485, służący do komunikacji za pomocą interfejsu RS485,
- ze względu na niedostępność oryginalnego regulatora napięcia, zastąpiono go układem MCP1701AT-5002I/CB, co wymusiło również zmianę kondensatorów w sekcji zasilania,
- z powodu konieczności zarezerwowania miejsca na złączu dla połączeń A i B interfejsu RS485, nie wyprowadzono lini SDA mikrokontrolera, co permanentnie uniemożliwia wykorzystanie TWI,
- dodano obsługę portu UART,
- utworzono namiastkę protokołu służącego do bezpiecznego zarządzania serwem,
- utworzono funkcje obsługujące komendy OpenServa.

Sposób kontrolowania zaprojektowanego OpenServa jest podobny do oryginalnego projektu. Różnice to konieczność stosowania interfejsu RS485 oraz zastosowanie odpowiedniego protokołu komunikacji z serwem. Podobnie jak w przypadku komunikacji przez TWI, wymagane jest wysłanie: adresu serwa, komendy/polecenia, zależnie od komendy, dodatkowo dwóch 16-bitowych słów danych (adresu rejestru,zapisywana wartość) oraz sumy CRC(modbus, polynomial 0xa001, wartość początkowa 0xffff), służącej do sprawdzenia poprawności odebranych danych.

1.4 Złącze modułu

Złącze modułu ma spełniać 3 funkcję: zasilania, komunikacji oraz programowania wewnętrznego mikrokontrolera.



Rysunek 1: Złącze modułu

Tabela 1: Opis wyprowadzeń modułu

Oznaczenie	Opis
JP1-1	Złącze zasilania 6V (10V Max) lub 5V w trybie programowania
JP1-2	Masa (GND)
JP1-3	SCK (programowanie)
JP1-4	Wyprowadzenie A interfejsu RS485
JP2-1	MOSI(programowanie)
JP2-2	MISO(programowanie)
JP2-3	RESET (programowanie)
JP2-4	Wyprowadzenie B interfejsu RS485

1.5 Komunikacja

Komunikacja z użyciem portu UART odbywa się przy prędkości transmisji (baudrate) równej 19200. Zgodnie z ideą interfejsu RS485, serwo jako urządzenie typu

slave oczekuje rozkazów, wysyłając dane tylko wtedy kiedy wysłane zostanie odpowiednie żądanie. W celu zapewnienia poprawności transmisji oraz jak najlepszej integracji z istniejącym firmwarem OpenServa zdecydowano się na stworzenie namiastki protokołu komunikacji. Komunikacja z serwem opiera się na wysyłaniu oraz ewentualnie odbieraniu ramek danych. Każda ramka składa się z 9 bajtów danych. Dzięki temu możliwe jest zaadresowanie 255 urządzeń (0=adres broadcast), zapiywanie i żadanie odczytania każdego z rejetrów za pomocą jednej ramki danych przy jednoczesnej gwarancji spójności i poprawności zapisanych/odczytanych danych. Sterowanie serwem opiera się na tej samej zasadzie jak w oryginalnym projekcie: zmiana pozycji, nastaw PID, lub jakichkolwiek innych ustawień następuje poprzez zapis danych (pozycji, nastaw) do odpowiednich rejestrów. Domyślny adres urządzenia to 0x10. Zmiana rejestru jest możliwa z poziomu kodu oraz przez zapis odpowiedniego rejestru. Rejestr pozycji serwa jest 16-bitowy, jed-

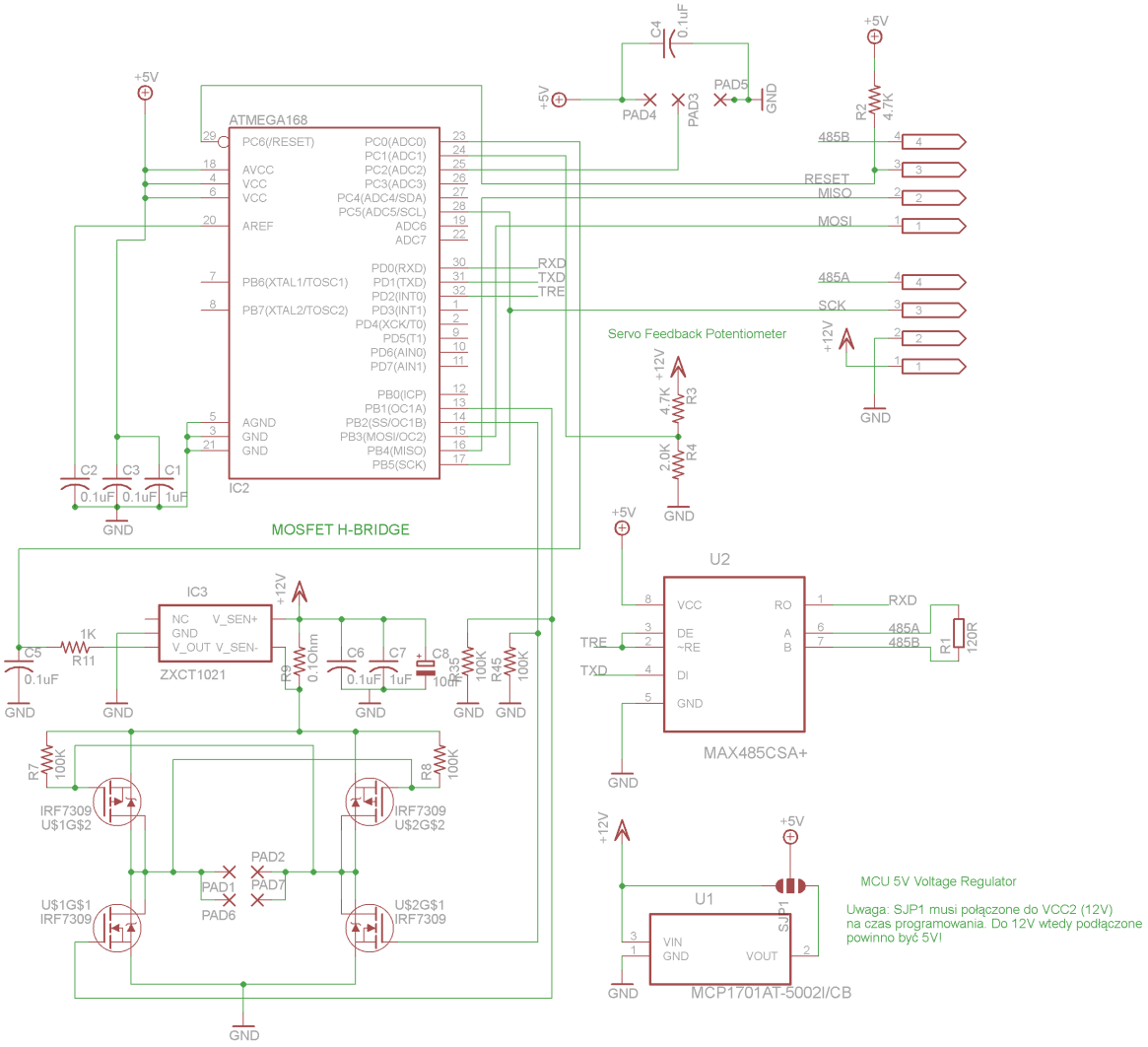
Tabela 2: Opis ramki danych

Zawiera:	Bajt:
1	'<'
2	adres urządzenia
3	polecenie
4	MSB data1
5	LSB data1
6	MSB data2
7	LSB data2
8	MSB CRC16
9	LSB CRC16

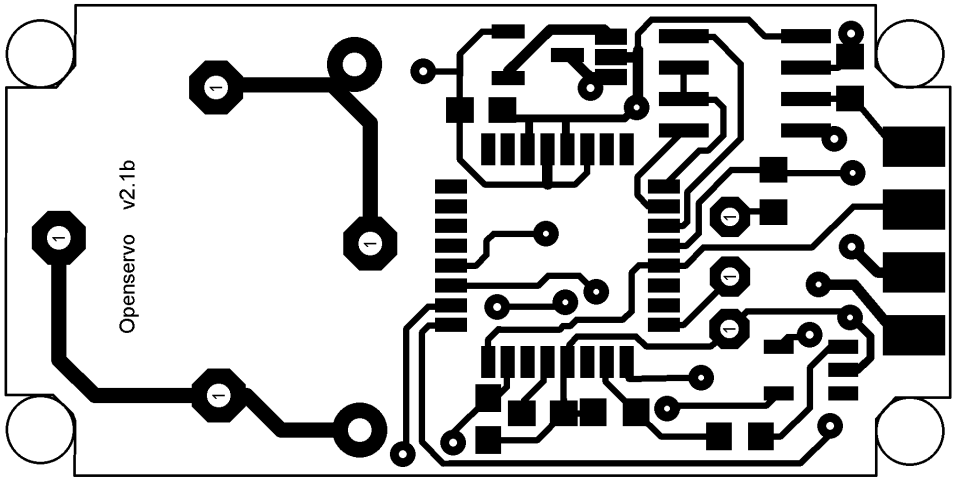
nak wykorzystywany zakres wartości to 0-1023. W OpenServie dodatkowo wprowadzona jest blokada mająca na celu ograniczenie zużycia serwa zawężające ten zakres do 96-928. Działanie wszystkich rejestrów jest tożsame z oryginalnym. Jedyną różnicą jest brak blokady zapisu istotnych rejestrów.

2 Hardware

2.1 Projekt



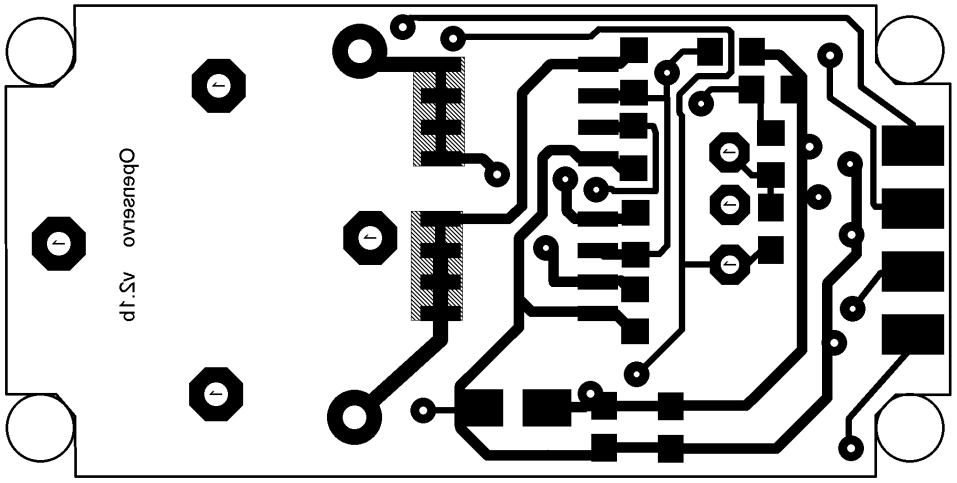
Rysunek 2: Schemat



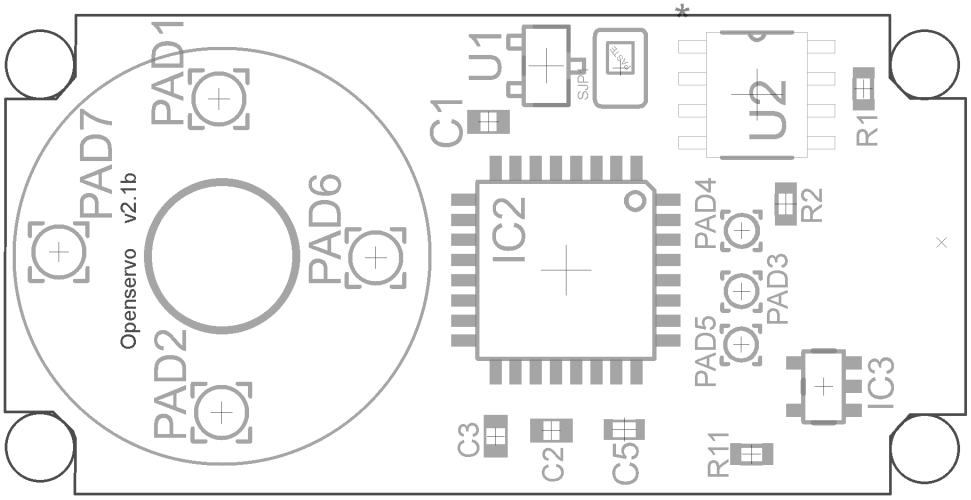
Rysunek 3: Projekt PCB (góra)

3 Oprogramowanie

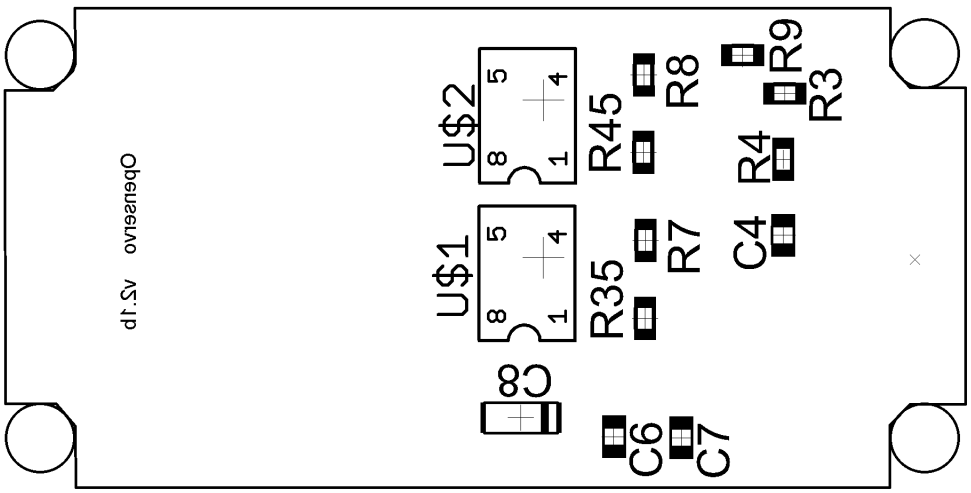
Firmware urządzenia opiera się głównie na oryginalnym oprogramowaniu OpenServa. Zostało ono odpowiednio rozszerzone aby spełniać cele i założenia projektu. Ponadto niezaimplementowana zostałą ochrona ważnych rejestrów - polecenia WRITE_ENABLE, WRITE_DISABLE nie mają efektu. Nastawy PID zostały dobrane metodą Zieglera-Nicholsa. Tabela rejetrów pobrana ze strony Projektu OpenServo:



Rysunek 4: Projekt PCB (dół)



Rysunek 5: Projekt PCB: rozmieszczenie części (góra)



Rysunek 6: Projekt PCB: rozmieszczenie części (dół)

Rysunek 7: Projekt PCB (góra)

Tabela 3: Rejestry OpenServa

Address	Name	Type	Description
0x00	DEVICE_TYPE	R/O	Device type - 1
0x01	DEVICE_SUBTYPE	R/O	Device subtype - 1
0x02	VERSION_MAJOR	R/O	Major version number
0x03	VERSION_MINOR	R/O	Minor version number
0x04	FLAGS_HI	R/O	Flags high byte
0x05	FLAGS_LO	R/O	Flags low byte
0x06	TIMER_HI	R/O	Timer high byte
0x07	TIMER_LO	R/O	Timer low byte
0x08	POSITION_HI	R/O	Servo position high byte
0x09	POSITION_LO	R/O	Servo position low byte
0x0A	VELOCITY_HI	R/O	Servo velocity high byte
0x0B	VELOCITY_LO	R/O	Servo velocity low byte
0x0C	POWER_HI	R/O	Servo power high byte
0x0D	POWER_LO	R/O	Servo power low byte
0x0E	PWM_CW	R/O	PWM clockwise value
0x0F	PWM_CCW	R/O	PWM counter-clockwise value
0x10	SEEK_HI	R/W	Seek position high byte
0x11	SEEK_LO	R/W	Seek position low byte
0x12	SEEK_VELOCITY_HI	R/W	Speed seek position high byte
0x13	SEEK_VELOCITY_LO	R/W	Speed seek position low byte
0x14	VOLTAGE_HI	R/W	Battery Voltage value high byte
0x15	VOLTAGE_LO	R/W	Battery Voltage value low byte
0x16	CURVE_RESERVED	R/W	reserved curve data
0x17	CURVE_BUFFER	R/W	Remaining curve buffer space
0x18	CURVE_DELTA_HI	R/W	Curve Time delta high byte
0x19	CURVE_DELTA_LO	R/W	Curve Time delta low byte
0x1A	CURVE_POSITION_HI	R/W	Curve position high byte
0x1B	CURVE_POSITION_LO	R/W	Curve position low byte
0x1C	CURVE_IN_VELOCITY_HI	R/W	Curve in velocity high byte
0x1D	CURVE_IN_VELOCITY_LO	R/W	Curve in velocity low byte
0x1E	CURVE_OUT_VELOCITY_HI	R/W	Curve out velocity high byte
0x1F	CURVE_OUT_VELOCITY_LO	R/W	Curve out velocity low byte
0x20	TWI_ADDRESS	R/W P	TWI address of servo
0x21	PID_DEADBAND	R/W P	Programmable PID deadband value
0x22	PID_PGAIN_HI	R/W P	PID proportional gain high byte
0x23	PID_PGAIN_LO	R/W P	PID proportional gain low byte
0x24	PID_DGAIN_HI	R/W P	PID derivative gain high byte
0x25	PID_DGAIN_LO	R/W P	PID derivative gain low byte
0x26	PID_IGAIN_HI	R/W P	PID integral gain high byte
0x27	PID_IGAIN_LO	R/W P	PID integral gain low byte
0x28	PWM_FREQ_DIVIDER_HI	R/W P	PWM frequency divider high byte
0x29	PWM_FREQ_DIVIDER_LO	R/W P	PWM frequency divider low byte
0x2A	MIN_SEEK_HI	R/W P	Minimum seek position high byte
0x2B	MIN_SEEK_LO	R/W P	Minimum seek position low byte
0x2C	MAX_SEEK_HI	R/W P	Maximum seek position high byte
0x2D	MAX_SEEK_LO	R/W P	Maximum seek position low byte
0x2E	REVERSE_SEEK	R/W P	Reverse seek sense
0x2F	RESERVED	R/W P	

Tabela 4: Komendy OpenServa

Command	Name	Description
0x00...0x7F	reserved	Reserved for data addresses (msb = 0)
0x80	RESET	Reset microcontroller
0x81	CHECKED_TXN	Read/Write registers with simple checksum
0x82	PWM_ENABLE	Enable PWM to motors
0x83	PWM_DISABLE	Disable PWM to servo motors
0x84	WRITE_ENABLE	Enable write of r/w protected registers
0x85	WRITE_DISABLE	Disable write of r/w protected registers
0x86	REGISTERS_SAVE	Save r/w protected registers fo EEPROM
0x87	REGISTERS_RESTORE	Restore r/w protected registers from EEPROM
0x88	REGISTERS_DEFAULT	Restore r/w protected registers to defaults
0x89	EEPROM_ERASE	Erase the AVR EEPROM
0x90	VOLTAGE_READ	Request a new Voltage sample
0x91	CURVE_MOTION_ENABLE	Enable curve based motion
0x92	CURVE_MOTION_DISABLE	Disable curve based motion
0x93	CURVE_MOTION_RESET	Clear the curve buffer
0x94	CURVE_MOTION_APPEND	Append a new curve