2020

Ročníková práce GVP

ROBOTICKÝ PAVOUK A OVLÁDÁNÍ

NEVOLE, VLK

Nevole, Vlk
GVP

Cíle projektu

Cílem našeho projektu bylo vyrobit a naprogramovat robotického pavouka a k němu udělat a naprogramovat ovládání. Chtěli jsme se při tom naučit ovládat servomotory a ovládat je bezdrátovým ovladačem. Zároveň jsme se rozhodli zjistit, jak nejlevněji můžeme takového robota postavit.

Realizace

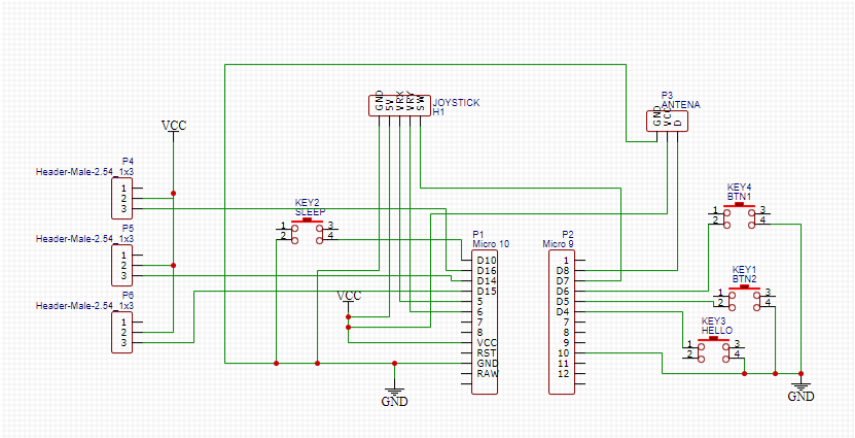
Jádrem celého projektu jsou oblíbené procesory od firmy Atmel Corporation, konkrétně Atmel ATmega328P a Arduino Pro Micro s čipem ATmega32U4.

Pro slave-master komunikaci jsme chtěli použít Bluetooth modul hc-05. Bohužel se nám nepodařilo navázat spojení mezi vysílačem a přijímačem. Proto jsme se rozhodlipro levnější vysílač a přijímač operující na frekvenci 433 MHz[1].V tomto případě jsme nakonec dali přednost ceně a jednoduchosti zapojení a programování. Bohužel i tato varianta má své nedostatky. Spojení není vždy spolehlivé a je pouze na omezenou vzdálenost, pak slábne a vysílač nepřijme vše, co vysílač vyšle.

Jako mechanický komponent, který hýbe celým pavoukem, jsme použili servomotory*TOWERPROSG-90 Micro*[2]. Díky poměru cena výkon jsou tyto serva velmi oblíbená. My jsme je použili hlavně kvůli jejich malým rozměrům22*11.5*27mm. Servomotory pracují na provozním napětí 3-7,2V, s momentem síly 0,016 kg*m.

Abychom nemuseli připojovat každé servo k vlastnímu pinu, použili jsme *IIC I2C Modulový driver servo motoru pro Arduino - PCA9685 16 kanálů 12-bit*[3] *PWM*, který je potřeba připojit k Arduinu pouze prostřednictvím pinů Data line(SDA) a clock line(SCL). K ovládání driveru jsme použili knihovnu *Adafruit_PWMServoDriver,* která nám umožnila ovládat serva pomocí příkazu setPWM(*pin na kterém je servo připojeno, počáteční poloha, úhel, na který chceme servo posunout*). Kdybychom nenapsali počáteční hodnotu nula, servo by prvně „poskočilo“ na inicializační úhel a až potom na konečný úhel.

Tištěný plošný spoj jsme vytvořili v prostředí EasyEDA. Z něj jsou také následující schémata zapojení obvodů.

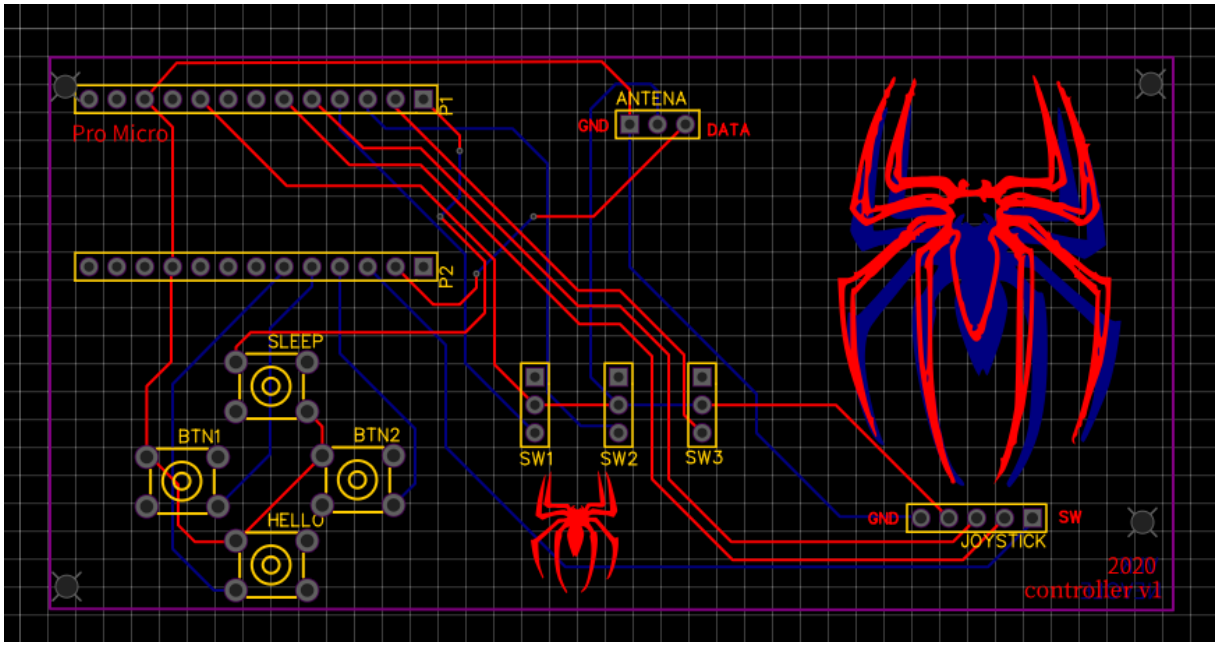


Obr. 1 Schéma ovladače

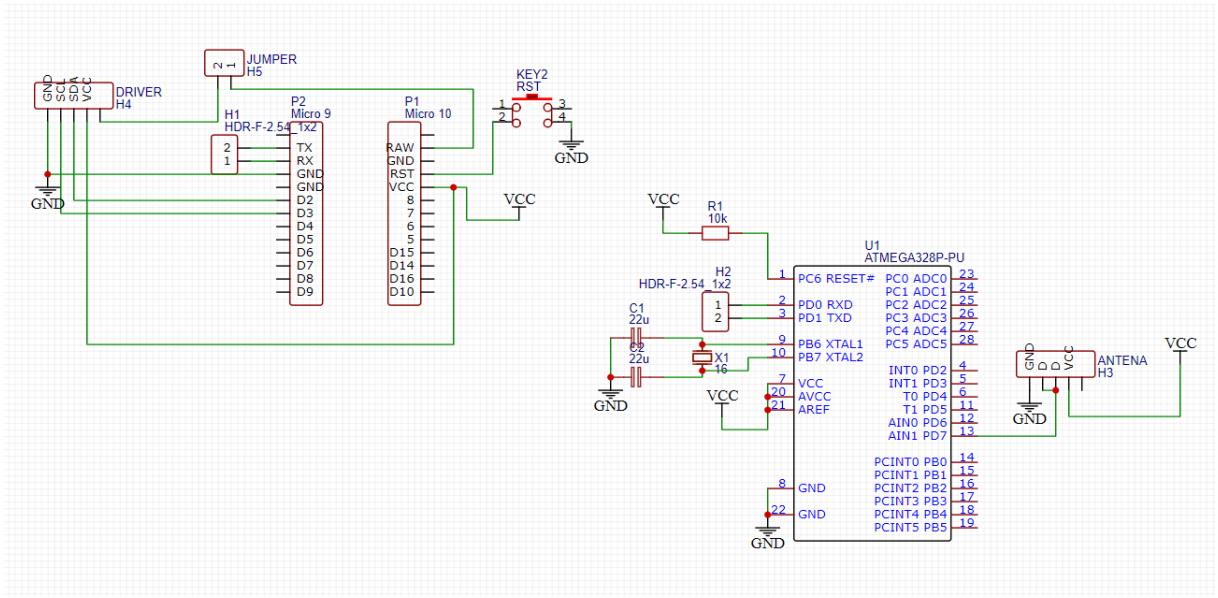
[1] Datasheet: https://arduino-shop.cz/docs/produkty/0/32/1427821401.pdf

[2]Datasheet: https://arduino-shop.cz/docs/produkty/0/741/eses1420669476.pdf

[3] Podrobnosti: https://arduino-shop.cz/arduino/1686-iic-i2c-modulovy-driver-servo-motoru-pro-arduino-pca9685-16-kanalu-12-bit-pwm.html

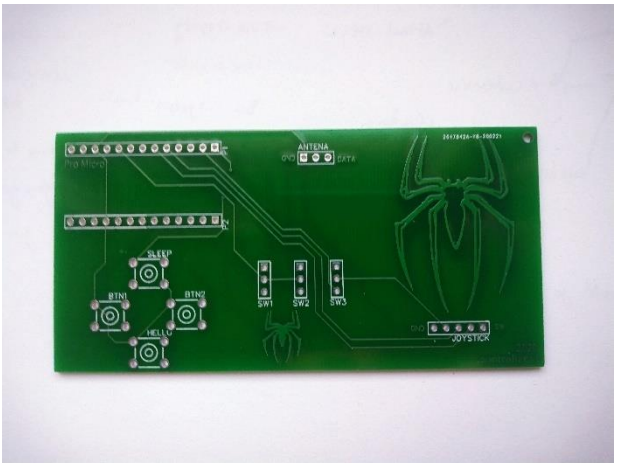


Obr. 2 Návrh plošného tištěného spoje ovladače

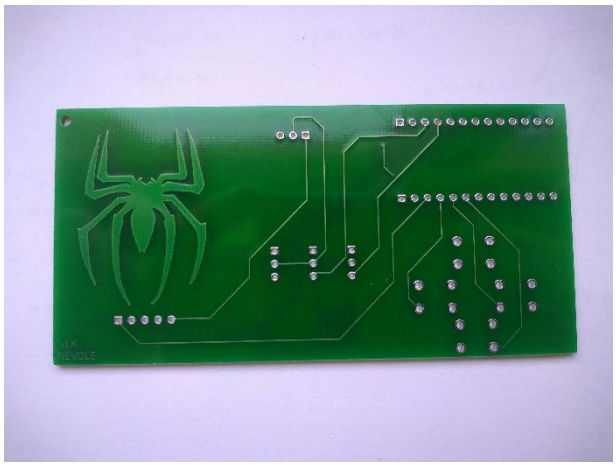


Obr. 3 Schéma přijímače a ovladače servomotorů

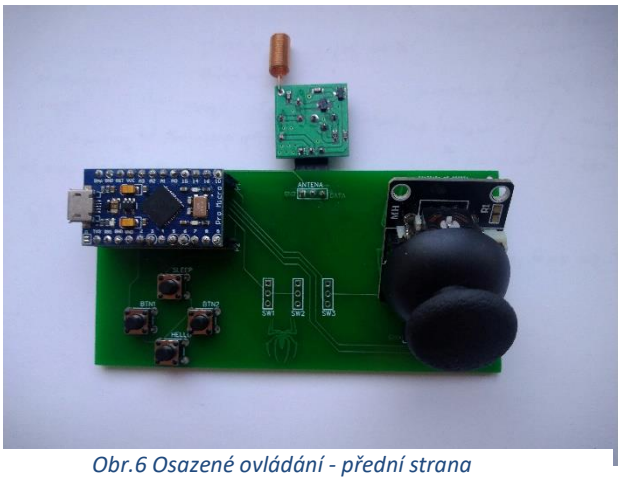
Samotný tištěný spoj jsme objednali u firmy JLCPCB. Rozhodli jsme se pro tuto variantu na základě předchozích zkušeností, spolehlivosti, kvalitě a nízké ceně. Výsledný tištěný spoj vypadá následovně.



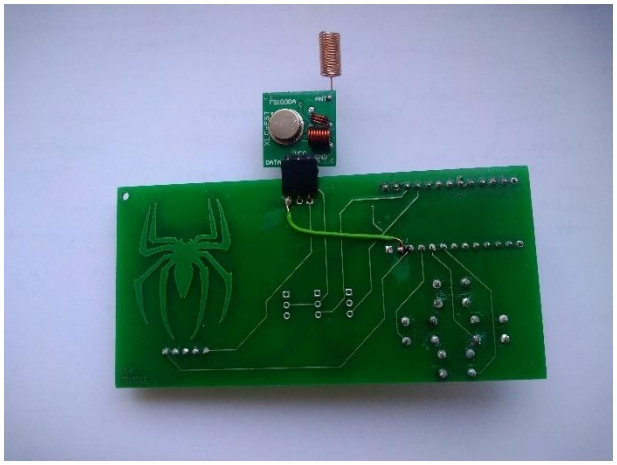
Obr.4 Tištěný plošný spoj ovládání - přední strana



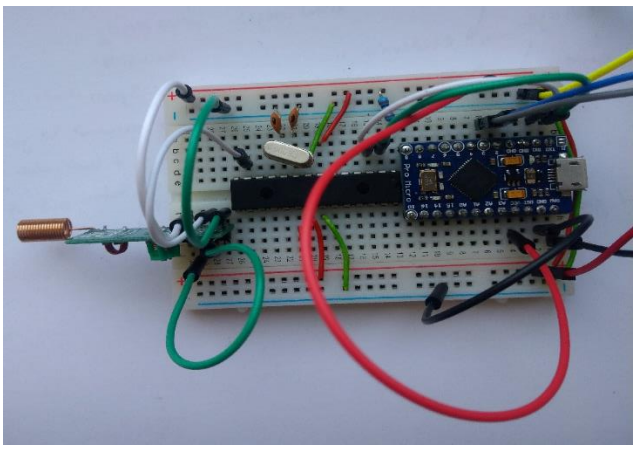
Obr. 5 Tištěný plošný spoj ovládání - zadní strana



Obr.6 Osazené ovládání - přední strana



Obr. 7 Osazené ovládání - zadní strana

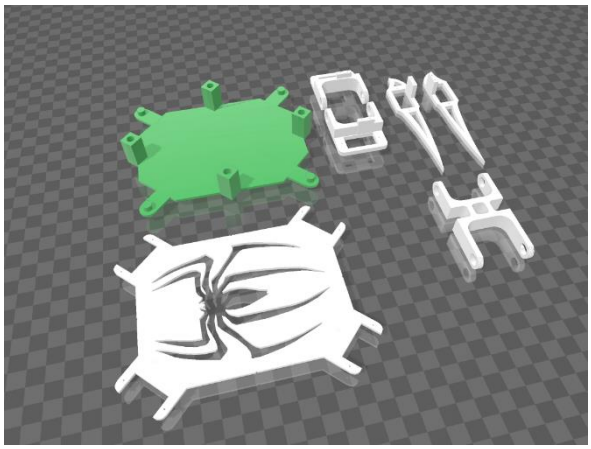


Obr. 8 Přijímač a ovládání servomotorů

Na spoji ovládání (obr. 7), na zadní části ovládacího panelu je připájený kabel. Je to z důvodu, že jsme na původním návrhu tištěného spoje nepropojili čip s anténou, takže nedocházelo k vysílání dat. Tento problém jsme již opravili. Návrh na obr. 2 již spojení obsahuje, na již objednaných spojích jsme museli připájet vodič externě.

Na obr. 8 je obvod pro příjem a ovládání servomotorů. Zatím je zapojený pouze v nepájivém kontaktním poli. Tištěný plošný spoj zatím nemáme.

Design těla (obr. 9) jsme zvolili pro jednoduchost a relevantně vysokou funkčnost. Velikost jsme přizpůsobili rozměrům servomotorů. Díly jsou vymodelované v programu 3D-builder a vytisknuté 3D-tiskárnou v ústřední knihovně na Mariánském náměstí.



Obr. 9 Design těla

Komplikace a řešení

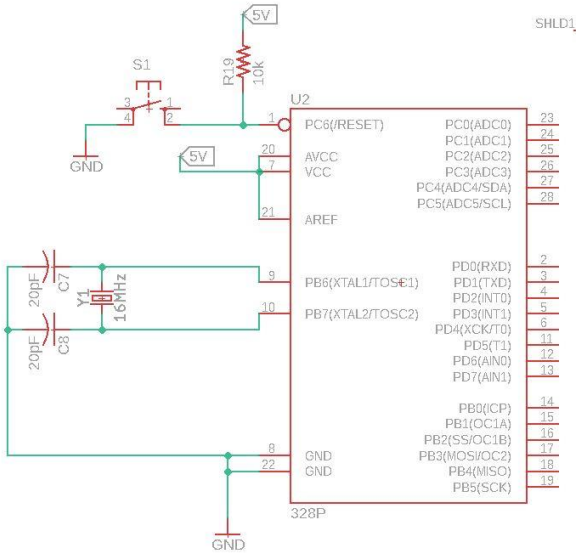
Při psaní programu pro ovládání pohybu jsme narazili na tři překážky. Ve chvíli, kdy se servomotor pohybuje, nemůže procesor vykonávat nic jiného. Aby pohyb vypadal plynule, bylo zapotřebí pohybovat všemi čtyřmi

nohami současně a to tak, že jedna noha udělá krok dopředu a ostatní tři se posouvaly zpátky. Museli jsme tedy pohyb rozdělit na postupné posouvání napsané pomocí for cyklu, což pohyb velice zpomalilo.

Druhý problém byl, že se servomotory nepohybují plynule, ale trhaně. Pokud by se měl posunout o větší kus, je pohyb potřeba rozdělit for cyklem na posun o menší úhly, což zapříčinilo další zpomalení.

Třetím a asi největším problémem bylo, že servodriver neumí zpracovat příkaz, otoč servo o jeden stupeň doleva. Knihovna, která ovládá servodriver, umí pouze otočit servo na nějaký úhel, který je stále stejný. Pokud chceme servem postupně otáčet, je potřeba pamatovat si úhel, na který je servo otočené. To byl velký problém při sestavování pavouka, protože jsme museli serva nastavit na nějaký výchozí úhel, který byl pro každou nohu jiný. Protože je to nejednotné, bylo potřeba napsat pro každou nohu vlastní pohybovou funkci.

Další komplikace nastala s ovládáním. Nepodařilo se nám efektivně napsat, aby procesor přijímal příkazy z ovládání, a zároveň komunikoval se servo driverem. Po konzultaci s panem doc. Ing. Janem Fischerem, CSc. Jsme se rozhodli pro přidání samostatného čipu na komunikaci s ovladačem. Na ten jsme použili procesor ATmega328P, který jsme zapojili podle schématu (obr. 10). Tento procesor přijímá signál z ovladače a předává ho po sériové komunikaci procesoru, který ovládá servomotory. Toto řešení znamenalo nárůst hmotnosti nákladu pro pavouka i zvýšení rozpočtu projektu, nicméně pro funkčnost bylo nezbytné.



Obr. 10 Zapojení ATmega328P

Program

Program je napsán v softwaru Arduino IDE, který podporuje nástavbu na C a C++. Celkem máme tři programy. Jeden, který je na ovladači a vyhodnocuje vstup uživatele (joystick a tlačítka) a vysílá jej. Další je na přijímání dat, a zároveň je předává pomocí sériové komunikace třetímu a poslednímu kódu, který podle nich hýbe se servomotory.

Na obr. 11 je funkce na pohyb zbylých třech nohou, které nedělají krok. Je potřeba je všechny průběžně posouvat dozadu, aby se cely pavouk posouval dopředu. Funkce se spouští společně s funkcí kroku dopředu.

```
360   for ( int angle = 700; angle > 430; angle--) {
361     delay(zpozdeni);
362     if (angle < 600) {
363       pwm.setPWM(9, 0, angle);
364     }
365     if (angle % 3 == 0) {
366       pwm.setPWM(8, 0, angle2 );
367       if (vpred){
368         angle2++;
369       }else{
370         angle2--;
371       }
372     }
373     if (angle % 3 == 0) {
374       pwm.setPWM(10, 0, angle3);
375       angle3--;
376     }
377     if (angle % 9==0){
378       if (turn){
379         if (vpred){
380           TurnL(CN);
381         }else{
382           TurnR(CN);
383         }
384       }else{
385         if (vpred){
386           Vposun(CN);
387         } else {
388           Zposun(CN);
389         }
390       }
391     }
392   }
393 }
```

Obr. 11 funkce pohybu

Funkce spánek (obr. 12) složí nohy do kompaktního tvaru. Je to funkce, která se spustí, když chceme pavouka vypnout. Dokud se servo nenastaví do požadované polohy, snaží se tam dotočit. Pokud tomuto pohybu něco brání, serva to ničí. Příkaz *setPWM(číslo serva, 0, 0)* servo zastaví na poloze, na které se zrovna nachází.

```
904 void spanek(){
905   if (Sleep == false){
906     pwm.setPWM(0 , 0 , 600);
907     delay(150);
908     pwm.setPWM(0 , 0 , 0);
909     pwm.setPWM(1 , 0 , 600);
910     delay(150);
911     pwm.setPWM(1 , 0 , 0);
912     pwm.setPWM(2 , 0 , 600);
913     delay(150);
914     pwm.setPWM(2 , 0 , 0);
915     pwm.setPWM(4 , 0 , 125);
916     delay(150);
917     pwm.setPWM(4 , 0 , 0);
918     pwm.setPWM(5 , 0 , 270);
919     delay(150);
920     pwm.setPWM(5 , 0 , 0);
921     pwm.setPWM(6 , 0 , 125);
922     delay(150);
923     pwm.setPWM(6 , 0 , 0);
924     pwm.setPWM(8 , 0 , 600);
925     delay(150);
926     pwm.setPWM(8 , 0 , 0);
927     pwm.setPWM(9 , 0 , 600);
928     delay(150);
929     pwm.setPWM(9 , 0 , 0);
930     pwm.setPWM(10 , 0 , 600);
931     delay(150);
932     pwm.setPWM(10 , 0 , 0);
933     pwm.setPWM(12 , 0 , 125);
```

Obr. 12 Funkce spánek

Podle toho, jakým směrem se pavouk pohybuje, se zavolá funkce (obr. 13) na pohybování zbylými třemi nohami. Parametr, který do ní zadáváme, je číslo nohy, která dělá krok vpřed.

```
577
578 void TurnR(int noha) {
579     switch (noha) {
580       case 1:
581         pwm.setPWM(0, 0, HipPos3);
582         pwm.setPWM(8, 0, HipPos4);
583         pwm.setPWM(12, 0, HipPos2);
584         if (HipPos3< 500){
585           HipPos3++;
586         }
587         if (HipPos4< 500){
588           HipPos4++;
589         }
590         if (HipPos2<435){
591           HipPos2++;
592         }
593         break;
594       case 2:
595         pwm.setPWM(0, 0, HipPos3);
596         pwm.setPWM(8, 0, HipPos4);
597         pwm.setPWM(4, 0, HipPos1);
598         if (HipPos3< 500){
599           HipPos3++;
600         }
601         if (HipPos4< 500){
602           HipPos4++;
603         }
604         if (HipPos1< 435){
605           HipPos1++;
606         }
```

Obr. 13 Funkce otočení vpravo

Výsledky práce a cíle do budoucna

Cílem této práce bylo sestavit pavouka připomínajícího robota, kterého bychom mohli ovládat pomocí bluetooth dálkového ovládání. Ačkoliv jsme ve finální verzi nepoužili komunikaci ve formě bluetooth, ale vysokofrekvenční vysílač a přijímač, cíl práce se nám splnit podařilo. Námi navržené tělo je funkční a software, který jsme napsali, funguje bez větších komplikací. Jediným problémem je, že občas přijímač nepřijme informace z vysílače a robot nereaguje. Kvůli současné situaci jsme včas nesehnali baterie na napájení pavouka a musí tedy chodit s kabelem.

Pokud bychom chtěli pokračovat ve vývoji robotů, považujeme toto za dobrý začátek. Jelikož práce na robotickém pavoukovi byla zábavná, mohli bychom se v budoucnu pokusit sestrojit vylepšenou verzi, Pavouk 2.0. Pro začátek bychom mohli vyměnit servomotory za výkonnější a napsat si vlastní knihovnu na jejich ovládán. Bylo by nutné zaměřit se na komunikaci a lépe si promyslet, jaké použijeme hardwarové komunikační rozhraní, aby bylo spolehlivější (alespoň do příště máme poučení, že ne na všem nevyplatí šetřit). Dále by bylo možné přidat různé tlakové a zvukové senzory a v neposlední řadě vymyslet umělou inteligenci na autonomní pohyb. Další cíle do budoucna jsou dodělat plošné spoje na tělo pavouka a optimalizovat ovládání. Věříme, že se nám je brzy povede splnit.