Térképező mobil robot fejlesztése

Somogyi Kornél

2020.01.28

Cél:

Egy olyan önálló robot megépítése, ami külső segítség nélkül képes térképet rajzolni egy ismeretlen helyiségről.

Fő hardverelemek:

- Raspberry Pi
- Arduino Pro Mini
- VL53L0X ToF távolságszenzorok
- NCR18650B Lítium akkumulátor
- A3967 léptetőmotorvezérlők
- 28BYJ-48 léptetőmotorok
- DC-DC konverterek

Feladatok:

- Váz tervezés és nyomtatás
- Összeszerelés
- Motorvezérlés és Távolságmérő szenzorok adatinak beolvasása Arduinoval
- Kommunikáció Arduino és Raspberry Pi közt
- Térképező algoritmus optimalizálása RPI-n
- Térkép rajzolás a beérkező adatok alapján

Váz

Az alapja a SMARS projekt, amely szabadon elérhető bárki számára. Mivel ez Arduinohoz készült, így ki kellett szélesíteni, illetve meghosszabbítani, a Micro USB portnál pedig kivágni, hogy a Raspberry beleférjen. VL53L0X szenzorhoz nem volt rögzítőelem, így azt nekem kellett tervezni. PLA műanyagból nyomtattam, mivel ez az anyag könnyen kezelhető, nagyon minimálisan vetemedik, nincs kellemetlen szaga mint az ABS-nek és olcsó.

Összeszerelés

4 szinten kerültek egymás fölé a nyomtatott áramkörök. A legalsó szinten van az 5 voltos tápfeszültség előállításához szükséges DC-DC konverter. Fölötte a motorvezérlőket találjuk, illetve a távolságszenzorokat. A motorok tápfeszültsége 8V, így ehhez is kellett egy DC-DC konverter, ami szintén itt található. A harmadik szinten a Raspberry Pi, a negyediken pedig egy protobard található. Ez utóbbira került az Arduino Pro Mini, egy 3.3 Voltos feszültségstabilizátor, illetve az akkumulátor és a töltőelektronika.

Motorvezérlés és Távolságmérő szenzorok adatainak beolvasása Arduinoval

A teljes rendszer 3.3 voltos jelszinten kommunikál. A motorvezérlők 2-2 vezetéken kommunikálnak. Az egyik a DIR, ami a forgásirányt határozza meg, ide csak magas vagy alacsony jelet kell küldeni. A másik a STEP, ami impulzusokat vár és értelemszerűen egy impulzus hatására 1 lépést tesz meg a DIR által meghatározott irányba. Ezt a sendpulses() függvény végzi.
A távolságmérő szenzorok szabványos I2C kommunikációt használnak, ezekhez az előre megírt Wire.h és a Pololu VL53L0X könyvtárát használtam.

Kommunikáció Arduino és Raspberry Pi közt

Azért volt szükség a 3.3 voltos rendszerfeszültségre, mert a Rasperry Pi a GPIO tüskesoron csak ezt a feszültséget fogadja el, nem 5V toleráns. A tüskesoron ugyan van 3.3V táp kivezetés, de ennek a maximális árama túl kevés lenne egy Arduino és a távolságszenzorok meghajtásához, ezért került fel a feszültségstabilizátor, ami már maximum 800 milliampert képes leadni, ez pedig bőven elég. A Pro Mini 16 Megahertzes órajelen 3.3 voltos tápon a gyártó szerint instabillá válhat, emiatt újraflasheltem a bootloadert és a fuse biteken átállítottam az órajelforrást a belső RC oszcillátorra, amelynek alapórajele 8MHz. Ehhez beállíthatunk további osztókat, hogy még alacsonyabb legyen, de ehhez a feszültséghez ennyi elég. Az alaplapról a kvarcot leforrasztottam, mert ez volt a legmagasabb alkatrész, túl közel volt a Raspberryhez és egyébként sem lesz már rá szükség. Az RC oszcillátor alacsonyabb fogyasztású, de pontatlanabb is. Ezt 115200-as baudrate mellett tökéletesen lehet érzékelni, szinte alig van értelmes adat, ami nem sérül meg. Emiatt 38400 fölötti baudrate-en a kommunikáció nehezen megvalósítható, oszcillátorkalibrálás nélkül. Én 9600 és 19200-on teszteltem, ilyen sebesség mellett tökéletesen stabil a kommunikáció, a Raspberry megkapja az adatokat. Először az impulzus hosszát adjuk meg neki, aztán azt, hogy melyik motort/motorokat forgassa és milyen irányba:

0 – jobb kerék előre
1 – jobb kerék hátra
2 – bal kerék hátra
3 – bal kerék előre
4 – jobb hátra, bal előre
5 – jobb előre, bal hátra
6 – mindkét kerék előre
7 – mindkét kerék hátra
8 – távolságszenzor kiolvasása

Ahhoz hogy a Raspberry Pi-n működjön a soros porton történő kommunikáció, be kell kapcsolnunk a raspi-configban, kikapcsolni a login shellt és a getty processzt leállítani. Ez utóbbit elvégzi a start.sh script, illetve indít egy VNC szervert.
SSH-n keresztül is tudunk ezután parancsokat kiadni a robotnak, a minicom szoftverrel.

Térképező algoritmus optimalizálása Raspberry Pi-n

Az algoritmus egy mohó legjobbat először keresés, amely nagy tömbök (esetünkben 120x120) vizsgálatánál képes nagyon lelassulni. Ezért csak a robot körüli részmátrixot figyeli a legtöbb esetben, csak akkor nézi a teljeset, amikor a közeli mátrixban nem talál felderítetlen pontot. Ezzel a megoldással a korábbi 0.5 fps helyett sikerült 25 fps körüli értéket kihozni az algoritmus szimulációjából, ami már elfogadhatónak mondható.

Térkép rajzolása a beérkező adatokból

Ez a feladat még nem készült el. A Processing felismeri az RPI soros portját, írni is lehet rá, azonban a bejövő adatokat valamiért nem tudtam vele beolvasni.

Használat:

Az akkumulátor mellett találunk egy bekapcsoló gombot. Indulás után az eszköz automatikusan csatlakozik a beállított Wifi hálózatra, SSH-n keresztül elérjük. Ezután a start.sh-t kell elindítani, majd minicommal vagy VNC-n keresztül Arduino IDE-vel 9600-as baudraten kiadni a parancsokat a ttyAMA0 portra.
Például: 3000 – Enter – 6 – Enter: 3000 egység mozgás előrefelé
1 – Enter – 8 – Enter: távolságszenzor adatainak beolvasása

Felmerült problémák, fejlesztési lehetőségek

- Akkumulátor rögzítés (az eredeti kontakthibákat okozott, a jelenlegi pedig gyakorlatilag nincs rögzítve)
- Váz újratervezése nagyobb alapterületűre, mivel ez a 4 szintes megoldás túl sok vezetéket igényel.
- Motorok cseréje NEMA17-re, mert a mostaniak fázisonként 150 milliamperes árammal is túlmelegszenek, a vezérlők pedig nem tudnak alacsonyabb áramot kiadni.
- Serial port problémák megoldása processingben