* Opcjonalne rzeczy do zrobienia:
  + zrzucić cały dysk tego co już jest
  + może sformatować kompa i ustawić na czysto ubuntu 16 + ROS kinetic? Tylko Dawid Kasieczka mówi, żeby najpierw zapytać dr Długosza, czy na pewno nie ma tam jakichś jego lokalnych zmian, które nie są na GIT-cie
* **(edytowane)**
* **[21:12]**Stack obecnie: kinetic i Ubuntu 16
* **[21:14]**Fajne info od twórcy projektu: Znane problemy z ADR:
  + Koła się ślizgają (dwa koła z taczek, chyba że ktoś je zmienił po czerwcu 2020) przy starcie/hamowaniu, chyba jest złagodzone przez SW teensy (Arduino-like płytka do kontroli IO)
* -Koła mają luzy, małą rozdzielczość enkoderów, brak kierunku w enkoderach (ogarnięte w teensy ale trzeba mieć w głowie)
  + Jedna z osi kół jest skrzywiona i powoduje że robot jadąc prosto twierdzi że jedzie po łuku
* Powyższe rzeczy sprawiają że nieważne od sposobu stawania na głowie odometria z kół będzie jaka będzie, czyli średnia w najlepszym wypadku. Warto żeby była jakaś, ale wg mnie nie da się jej dużo polepszyć niż to co napisałem na teensy i moim zdaniem bez zmiany HW nie warto. Ponadto: -Dane z czujników ultradźwiękowych są mało użyteczne i ich integrację zostawić na sam koniec. -Dane z GPSa i ich integracja również na sam koniec, istotna jest odometria +lidar lub odometria + imu + lidar.
  + Dane z Kinecta są użyteczne ale niekoniecznie potrzebne na początku.
* **[21:14]**Od czego zacząć (sporo powinno być gotowe w repo): -Dowiedzieć się jaki ROS jest w repo oraz na komputerze głównym. W 2020 był to ROS Kinetic i Ubuntu 16, ale mogło się to zmienić.
  + Lidar i teensy powinny mieć udev rules w Linuxie, tak aby meldowały się jako np. /dev/lidar a nie /dev/ttyUSB0 żeby uniknąć konfliktów nazw i problemów np. z kolejnością wpinania urządzeń USB
* -Sieć powinna być dobrze skonfigurowana, tak aby po podłączeniu do rotuera który jest na robocie można było odbierać i wysyłać ros msgs (jak działa tylko jedno albo drugie to brakuje konfiguracji gdzieś) (znany problem: robot bez routera na sobie łączył się do stacjonarnej sieci, jak miał ROS\_MASTER albo ROS\_IP (nie pamiętam) ustawione na 192.168… to ROS się wysypywał po odjechaniu poza zasięg, natomiast gdy było na loopback, to nie było komunikacji z zewnątrz) -launch który odpala LIDAR, komunikuje się z teensy i widoczne są poprawne dane z lidara, odometrii, potem IMU (nie pamiętam gdzie jest wpięte), poprawny TF, joystick Xbox 360 steruje robotem (topic cmd\_vel). !!Upewnić się 10 razy że każdy msg ma timestamp, poprawne frame\_id, poprawne dane (wyskalowane) i poprawny topic!! -Zweryfikować odometrię (pokazuje dobry kierunek i poprawnie skręca plus jest wyskalowana poprawnie na tyle ile się da)
* **[21:14]**-Jeśli jest taki launch i odometria jakoś działa, to zgrać kilka przejazdów które posłużą do dobrania parametrów tworzenia mapy (rosbagi z odometrią, IMU, LIDARem, zweryfikować czy dane mają sens). Na podstawie tych rosbagów zrobić skrypt który offline symuluje przejazd w tempie np. 5x, odpalać gmapping i dobierać parametry aż mapa będzie dobra (przy czym użyć wielu rosbagów żeby uniknąć „overtrainingu”). Dobieranie parametrów na żywym robocie i ręczne powtarzanie jazd to proszenie się o zmarnowanie kilkudziesięciu godzin i niezrobienie niczego, absolutnie niezalecane.o
  + Jeśli robienie mapy działa to można robić autonomiczną jazdę, temat jest szeroki, warto rzucić okiem na to co jest w repo. Ponadto tutaj jest poradnik:<https://wiki.ros.org/navigation/Tutorials>. Jeśli program nie generuje danych/nie odpowiada to 90% nie dostaje danych lub dostaje złe dane.
* **[21:15]**<https://gitlab.com/knintegra/adr-i->