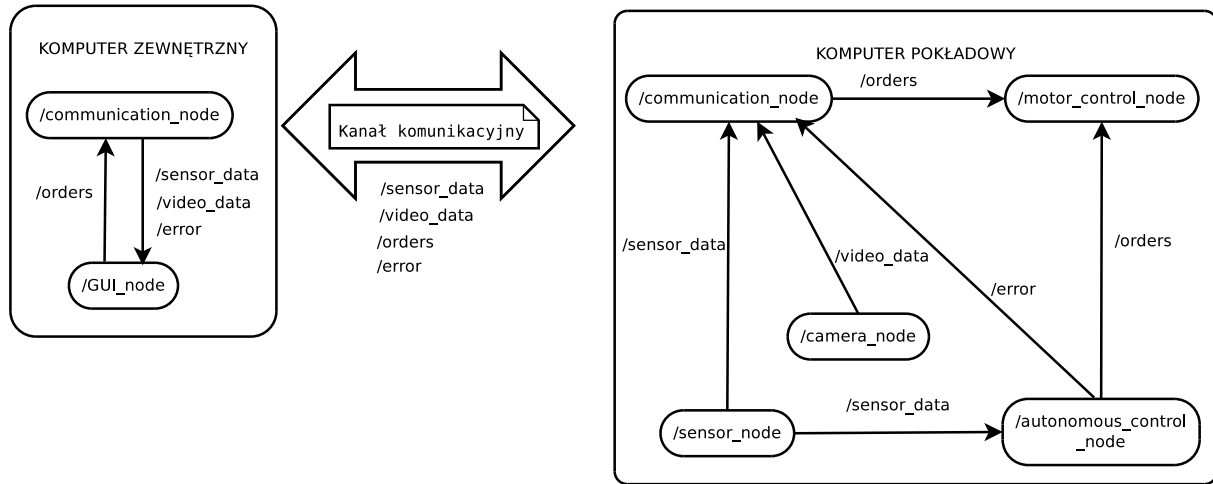


# Architektura oprogramowania

## 1 Diagram węzłów



Rysunek 1: Diagram komponentów programowych przedstawionych jako węzły i tematy ROSa

Powyższy rysunek jest rozwinięciem diagramu komponentów programowych (założenia projektowe, rysunek 2), uwzględniającym szczegóły implementacyjne na poziomie pojedynczych węzłów i tematów ROSa. Tematy oznaczone są jako nazwy nad strzałkami łączącymi komponenty. Służą one do przekazywania między węzłami konkretnego typu danych. Typy danych opisane są szczegółowo w sekcji 3.

## 2 Opis węzłów

Nazwa węzła	Opis	Subskrybowane tematy	Publikacje
/communication_node	Zajmuje się transmisją danych pomiędzy maszynami. Analogiczny węzeł działa na komputerze zewnętrznym, przy odwrotnym stanie tematów subskrybowanych/publikacji	/sensor_data /video_data /error	/orders
/sensor_node	Publikuje informacje z czujników podłączonych do GPIO Raspberry Pi	-	/sensor_data
/camera_node	Publikuje dane z kamery	-	/video_data
/autonomous_control_node	Podejmuje decyzje o awaryjnym zatrzymaniu robota. Gdy taka sytuacja nastąpi, przesyła odpowiednią informację użytkownikowi.	/sensor_data	/orders /error
/motor_control_node	Na podstawie otrzymanych rozkazów generuje sygnały sterujące silnikami	/orders	-
/GUI_node	Stanowi interfejs dla użytkownika. Pozwala na wydawanie rozkazów oraz podgląd obrazu z kamery i czujników	/sensor_data /video_data /error	/orders

Tabela 1: Lista głównych komponentów programowych oraz ich danych wejściowych i wyjściowych

Węzły są najbardziej intuicyjną reprezentacją komponentów programowych. Są to równolegle działające procesy wykonujące wyspecjalizowane zadania. Dzięki mechanizmowi subskrybowania i publikowania wiadomości w tematach, można w łatwy i wyraźny sposób określić ich dane wejściowe i wyjściowe. Zgodnie z dobrą praktyką programistyczną, węzły nazwane są w języku angielskim.

### 3 Opis typów danych

Zgodnie z mechanizmem systemu ROS, każdy temat ma przypisany swój konkretny typ danych. Poniżej znajduje się lista typów używanych w projekcie. Większość z nich to typy złożone. Zagnieżdżenie typów - zgodnie z konwencją stosowaną w ROSie - oznaczone jest poprzez wcięcie. Jako że niektóre typy danych muszą być skrojone dokładnie na miarę robota (np. dane z czujników), utworzona zostanie specjalna paczka `LIGO_msgs`, w której będą one zawarte.

- Temat `/video_data`

```
sensor_msgs/Image video
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

- Temat `/sensor_data`

```
LIGO_msgs/Sensor_data data
uint8 left_sensor
uint8 mid_sensor
uint8 right_sensor
bool contactor1
bool contactor2
bool contactor3
bool contactor4
bool contactor5
bool contactor6
bool contactor7
bool contactor8
```

- Temat `/orders`

```
LIGO_msgs/Order_data order
int8 left_velocity
int8 right_velocity
bool priority
```

- Temat `/error`

```
int8 error_code
```

### 4 Podział obowiązków i podsumowanie

Za wszystkie komponenty wchodzące w skład oprogramowania komputera pokładowego odpowiedzialny jest Piotr Dulewicz. Napisaniem programów działających na komputerze zewnętrznym zajmuje się Piotr Jabłoński. Komunikacją zajmuje się Andrzej Szmyt.

Dzięki dokładnie zdefiniowanym typom danych system jest spójny, i połączenie kilku niezależnie tworzonych części oprogramowania nie powinno sprawiać problemów.