# LL4 – Model Car

Florian Mauracher

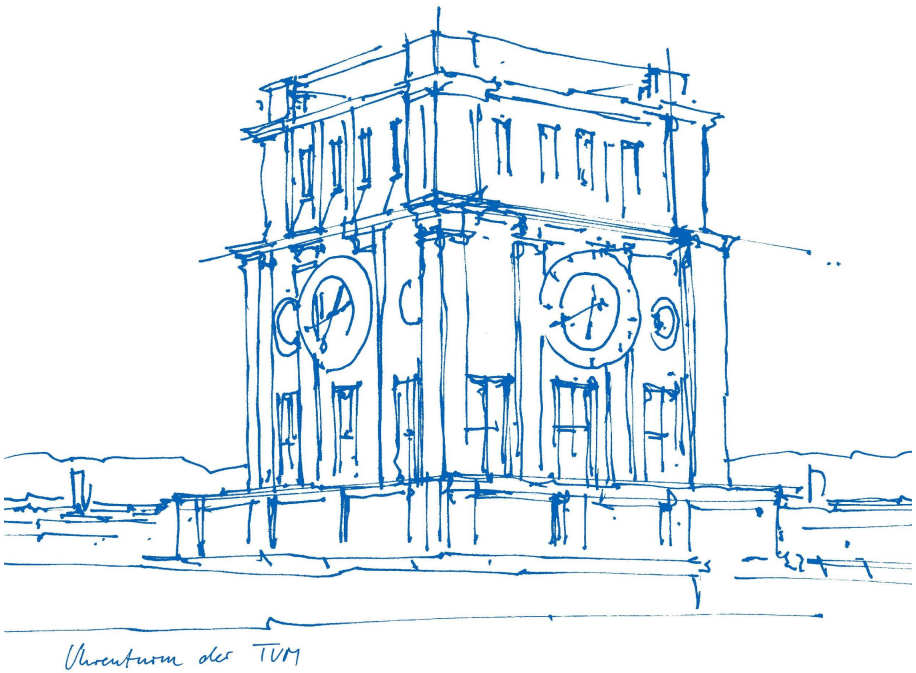
Christoph Griesbeck

Thomas Mauerer

Technische Universität München

Department of Computer Science

Garching, 01. Aug, 2017



# Goals of the project

**Originally planned:**

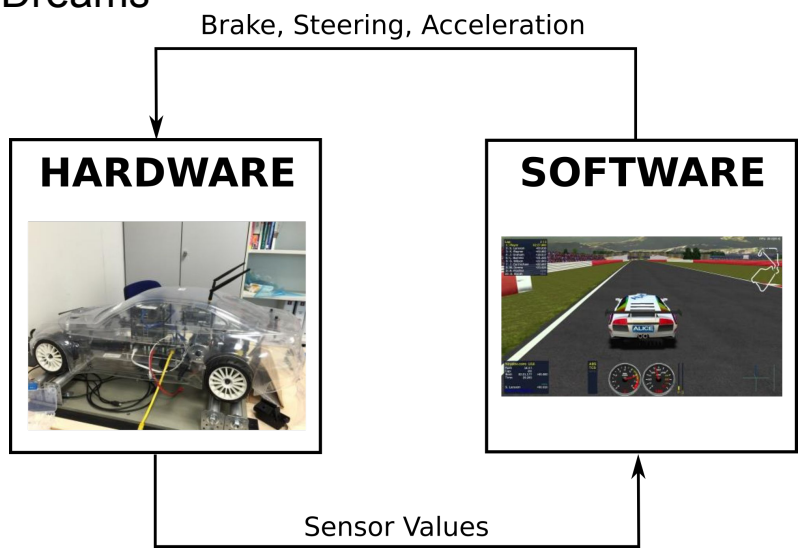
- Hardware in the loop in combination with SpeedDreams
- Genode with Fiasco.OC

**Our part:**

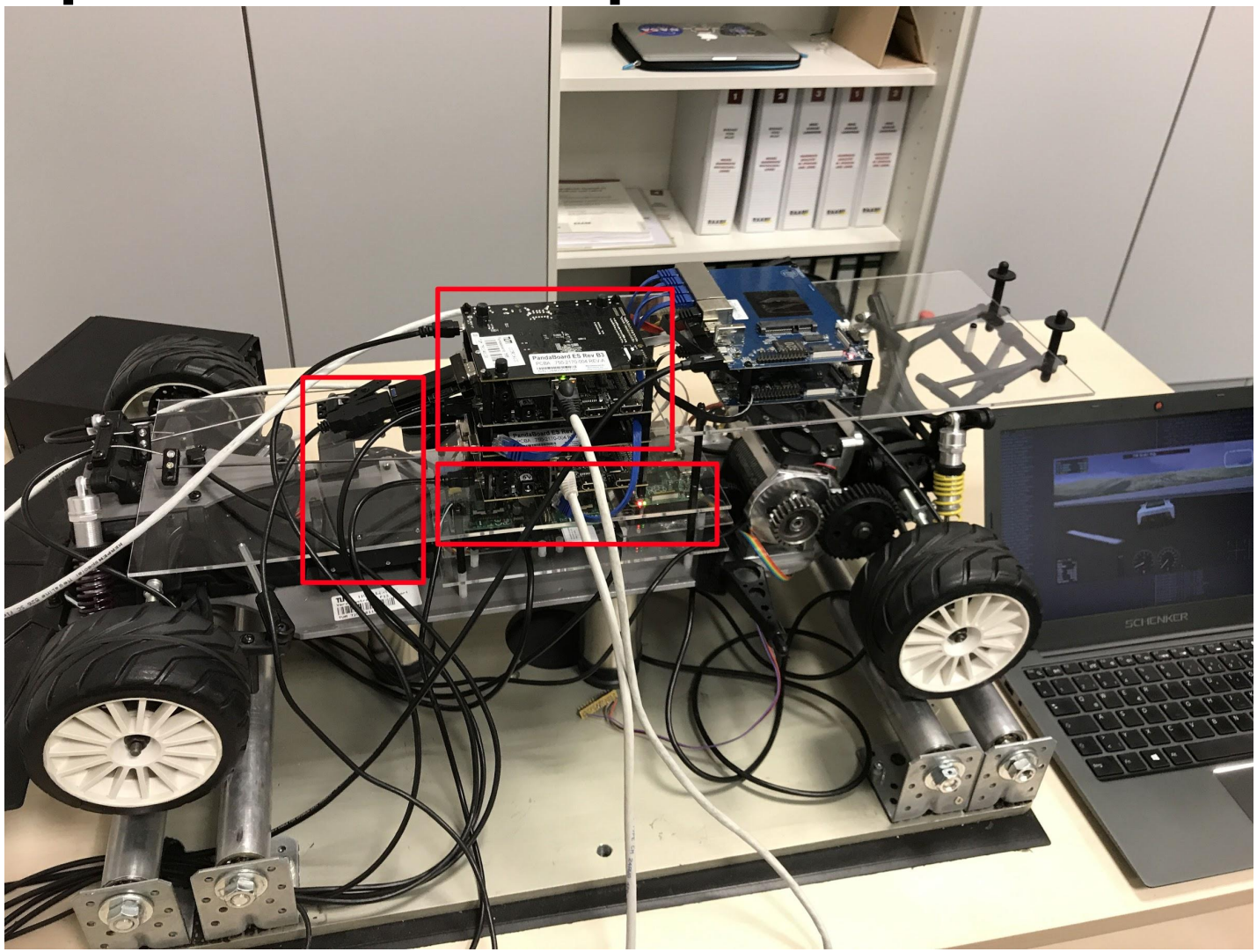
- Interpretation of simulation data from SpeedDreams
- Actuation of servos in the model car

**Problem:**

- No sensors in the model car
- Therefore, loop not closed



# Description of the setup



# 1. Servos

**Servos:**

- 3 Braking Servos
- 1 Steering Servo
- 1 Engine


**Tasks:**

- Control braking and steering servos
- Control engine (optional)

# 1. Servos

**Servos:**

- 3 Braking Servos
- 1 Steering Servo
- 1 Engine


**Tasks:**

- Control braking and steering servos ✔
- Control engine (optional) ✘

# 2. Servo Controller Board

**Pololu Maestro Controller Board**

- 12 channels
- Controls servos via pwm signals
- Serial protocol

**Tasks:**

- Connect servos to the controller board
- Commissioning of the servo controller board
- Check functionality

# 2. Servo Controller Board

**Pololu Maestro Controller Board**

- 12 channels
- Controls servos via pwm signals
- Serial protocol

**Tasks:**

- Connect servos to the controller board ✔
- Commissioning of the servo controller board ✔
- Check functionality ✔

# 3. Raspberry Pi


**Tasks:**

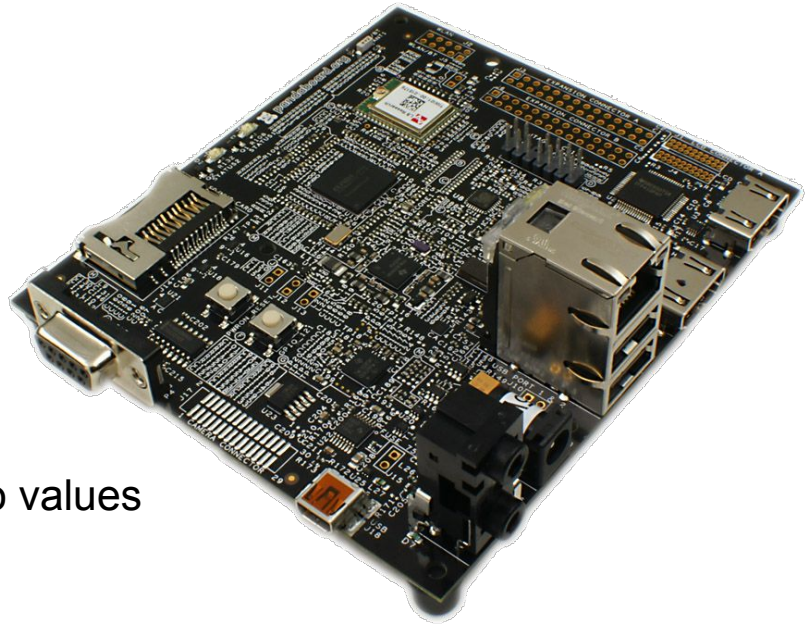
- Install Genode with Fiasco.OC
- Implement ProtoBuf by Google
- Develop MQTT client
- Connect with servo controller board
- Implement Genode Servo controller application
- Convert control commands into concrete servo values
- Read sensor values (optional)

# 3. Raspberry Pi


**Tasks:**

- Install Genode with Fiasco.OC ✔
- Implement ProtoBuf by Google ✘
- Develop MQTT client ✔
- Connect with servo controller board ✔
- Implement Genode Servo controller application ✔
- Convert control commands into concrete servo values ✘
- Read sensor values (optional) ✘

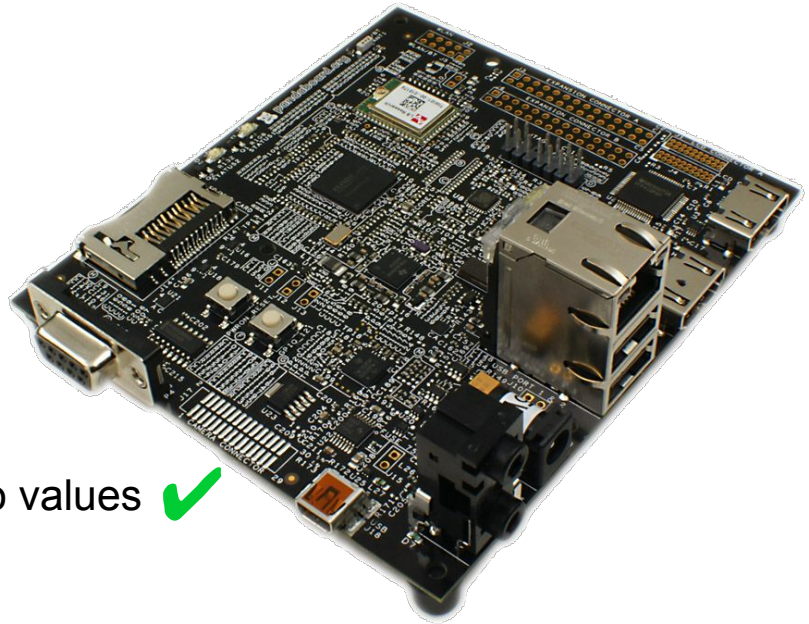
# 4. PandaBoard



**Tasks:**

- Install Genode with Fiasco.OC
- Develop MQTT client
- Generate control commands
- Convert control commands into concrete servo values

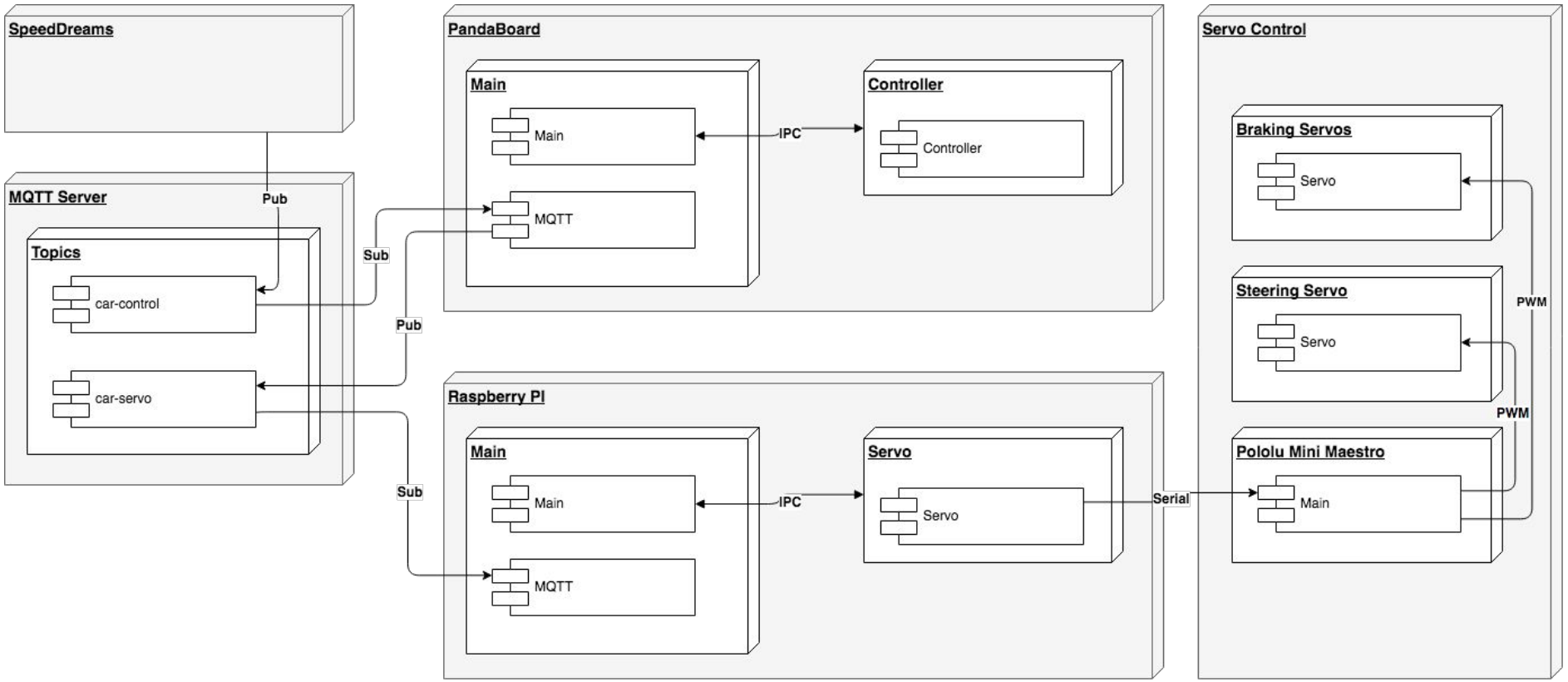
# 4. PandaBoard



**Tasks:**

- Install Genode with Fiasco.OC ✔
- Develop MQTT client ✔
- Generate control commands ✘
- Convert control commands into concrete servo values ✔

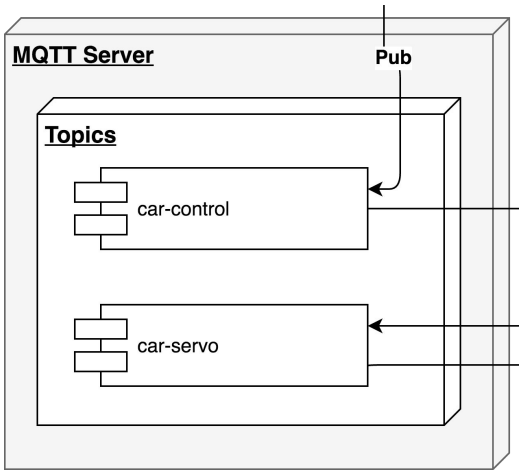
# Component overview



# MQTT

**Simulation → PandaBoard**

- Topic: car-control
- Format: (command,value)
- Example: (1,0.5)

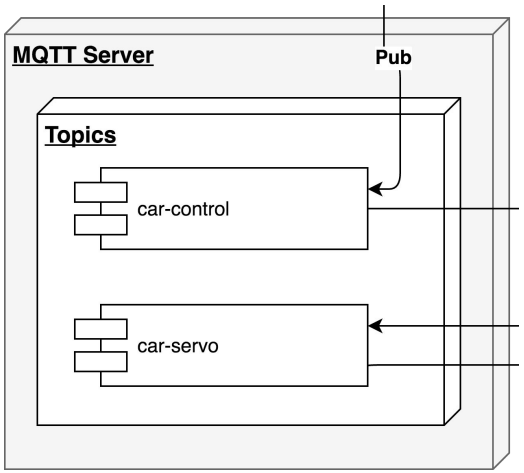



| Command | Value Range | Meaning |
|---|---|---|
| 0 | [ -1.0 ; 1.0 ] | Steering |
| 1 | [ 0 ; 1.0 ] | Brake |
| 2 | [ 0 ; 1.0 ] | Acceleration |

# MQTT

**PandaBoard → Raspberry Pi**

- Topic: car-servo
- Format: (channel,value)
- Example: (0,7500)



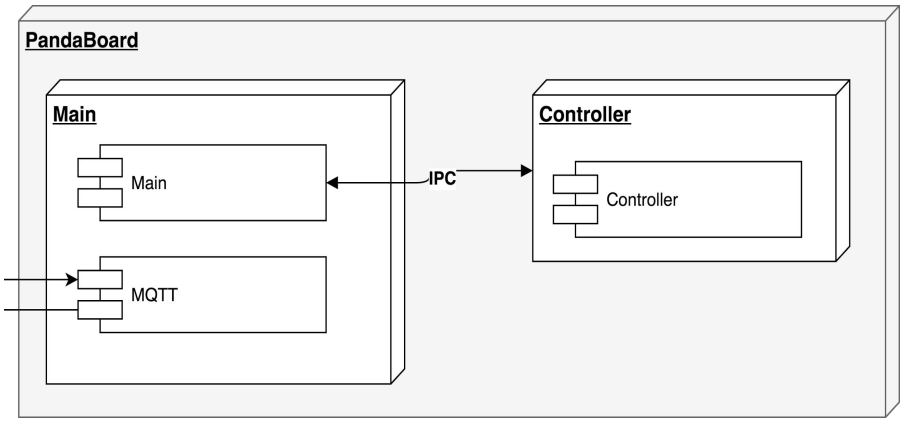

**Channel:**

- 0,1,2: Braking servos
- 6: Steering Servo

**Values:**

- 4500 – 7500
- 6000: neutral

# PandaBoard




**Genode Tasks:**

- Main component
- Controller component

**Main component:**

- Setup network
- Subscribe to car-control mqtt topic
- Receive mqtt messages
- Call controller
- Publish message to car-servo mqtt topic

**Controller component:**

- Transforms control commands into concrete servo values

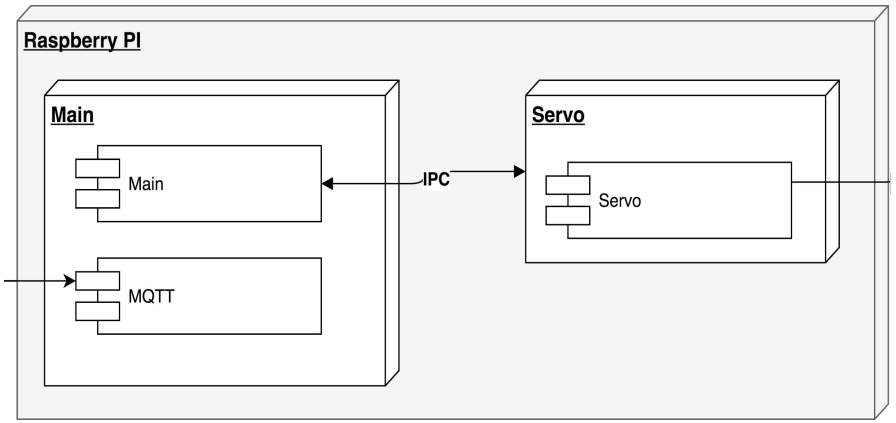
# PandaBoard – Code Snippets

```
while (true) {
    mqtt_entity->wait_msg();
    mqtt_entity->get_cmd(recv_cmd, sizeof(recv_cmd));

    split = strtok(recv_cmd, ",");
    id = split;

    target = strtok(NULL, ",");
    value = atof(target);

    switch (strtoul(id, NULL, 0)) {
        case STEER:
            servoVal = controller.transform_steer(value);
            snprintf(servo_cmd, sizeof(servo_cmd), "%s,%d", STEER_CHANNEL, servoVal);
            mqtt_entity->send_message(servo_cmd);
            break;
        case BRAKE:
            servoVal = controller.transform_brake(value);
            snprintf(servo_cmd, sizeof(servo_cmd), "%s,%d", BRAKE_LEFT_FRONT_CHANNEL, servoVal);
            mqtt_entity->send_message(servo_cmd);
            snprintf(servo_cmd, sizeof(servo_cmd), "%s,%d", BRAKE_RIGHT_FRONT_CHANNEL, servoVal);
            mqtt_entity->send_message(servo_cmd);
            snprintf(servo_cmd, sizeof(servo_cmd), "%s,%d", BRAKE_REAR_CHANNEL, servoVal);
            mqtt_entity->send_message(servo_cmd);
            break;
    }
}
```

# PandaBoard – Code Snippets

```
int transform_steer(double value) {
    if (value < -1 || value > 1) {
        PERR("Invalid steering angle - range is -1 to 1");
        return -1;
    }

    // Invert value as SpeedDreams thinks -1 is right
    value = -value;

    value = (value + 1)/2;
    return (SERVO_UPPER_BOUND - SERVO_LOWER_BOUND) * value + SERVO_LOWER_BOUND;
}

int transform_brake(double value) {
    if (value < 0 || value > 1) {
        PERR("Invalid target brake position - range is 0 to 1");
        return -1;
    }

    return (SERVO_UPPER_BOUND - SERVO_LOWER_BOUND) * value + SERVO_LOWER_BOUND;
}
```

# Raspberry Pi




**Genode Tasks:**

- Main component
- Servo component

**Main component:**

- Setup network
- Subscribe to car-servo mqtt topic
- Receive mqtt messages
- Call servo

**Servo component:**

- Sends command to servo controller board

# Raspberry Pi – Code Snippets

```
int setTarget(unsigned char channel, unsigned short target) {

    if (channel > 11) {
        PERR("Channel does not exist");
        return -1;
    }

    if (target < 4000 || target > 8000) {
        PERR("Invalid target position - range is 4000 to 8000");
        return -1;
    }

    unsigned char command[] =
{0x84, channel, (unsigned char)(target & 0x7F), (unsigned char)(target >> 7 & 0x7F)};
    if (_terminal->write(command, sizeof(command)) < sizeof(command)) {
        PERR("error writing");
        return -1;
    }
    return 0;
}
```

# Problems & Difficulties

- Initial build environment
  - Lab computer problems
  - TFTP boot not working
- Raspberry Pi build
- Broken steering servo
- Genode complex

# Demo Video