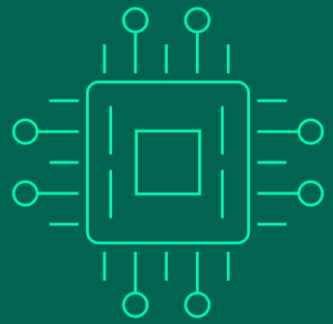
sea|me

software engineering in automotive
and mobility ecosystems

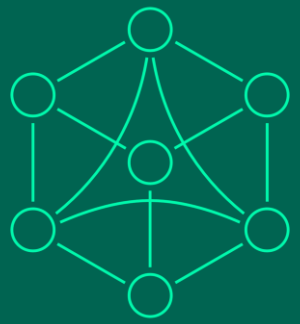
# DES 2: INSTRUMENT-CLUSTER PRESENTATION

<Team 5>

DongHwan Seo

Mounika Vinjarapu



Sujong Ha


18.08.23

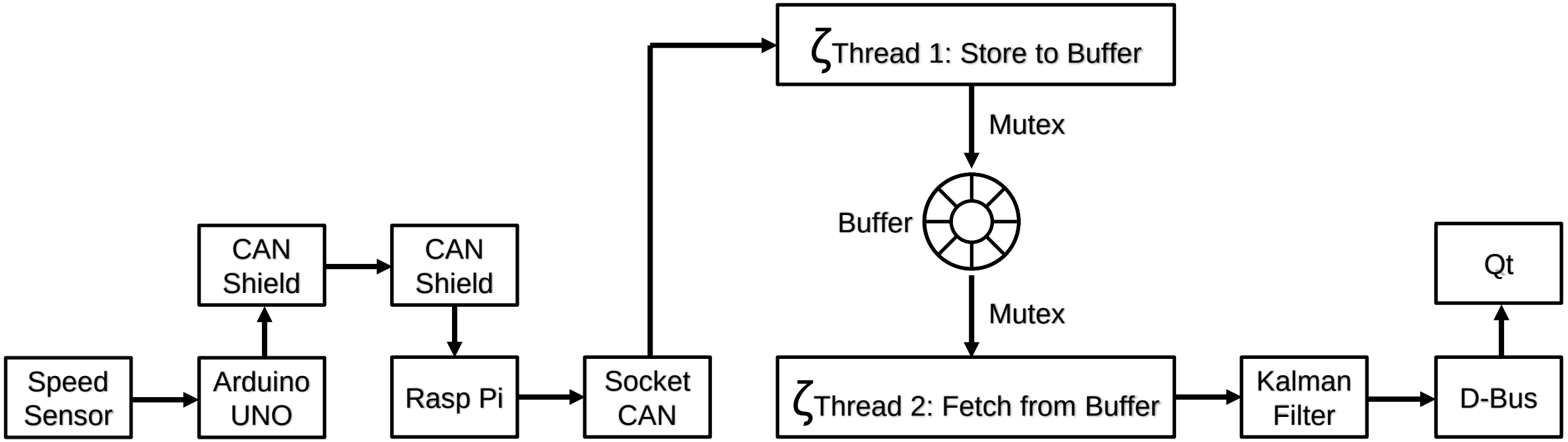
# INTRODUCTION

sea|me
software engineering in automotive and mobility ecosystems

The main goal of the **Instrument-Cluster** project is to create a functioning dashboard for a PiRacer car that displays **real-time** speed data & battery level via **CAN bus**.

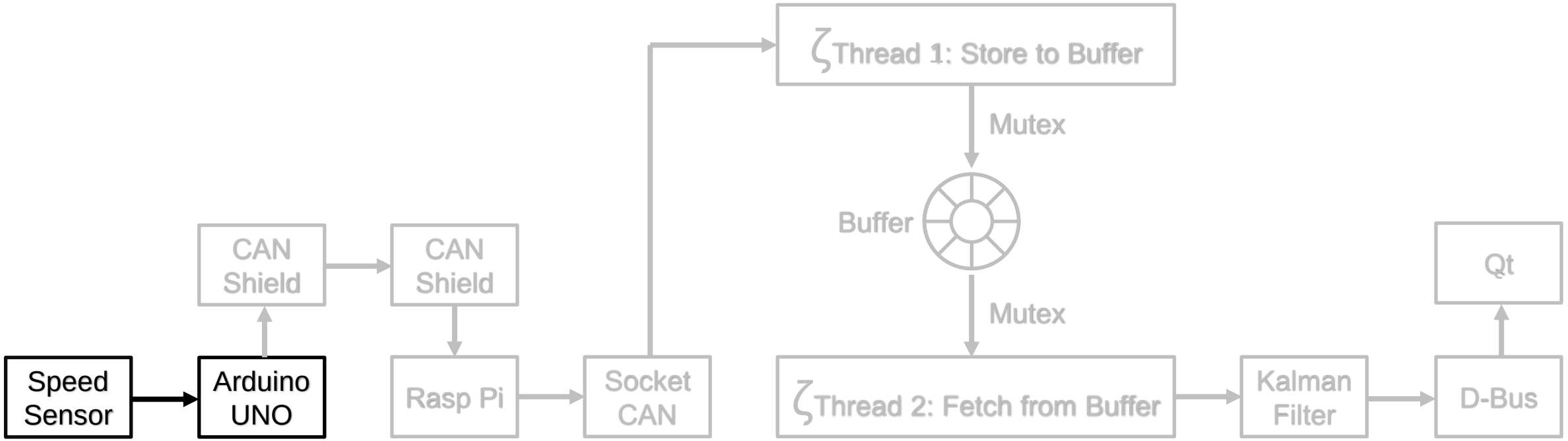
# INDEX

sea|me
software engineering in automotive and mobility ecosystems



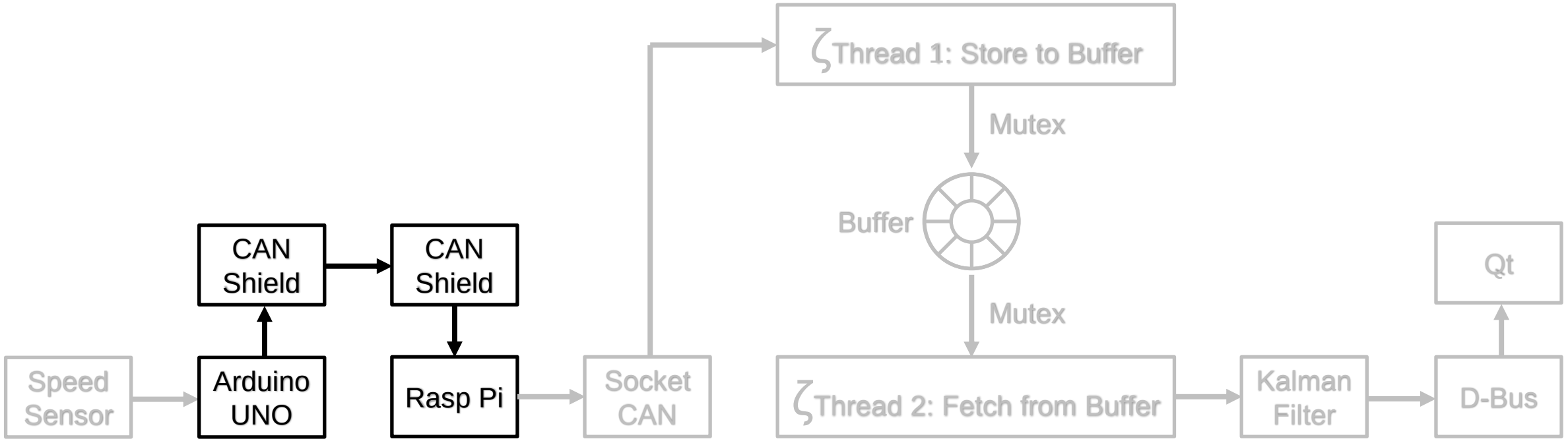
# INDEX

1. Speed Sensor



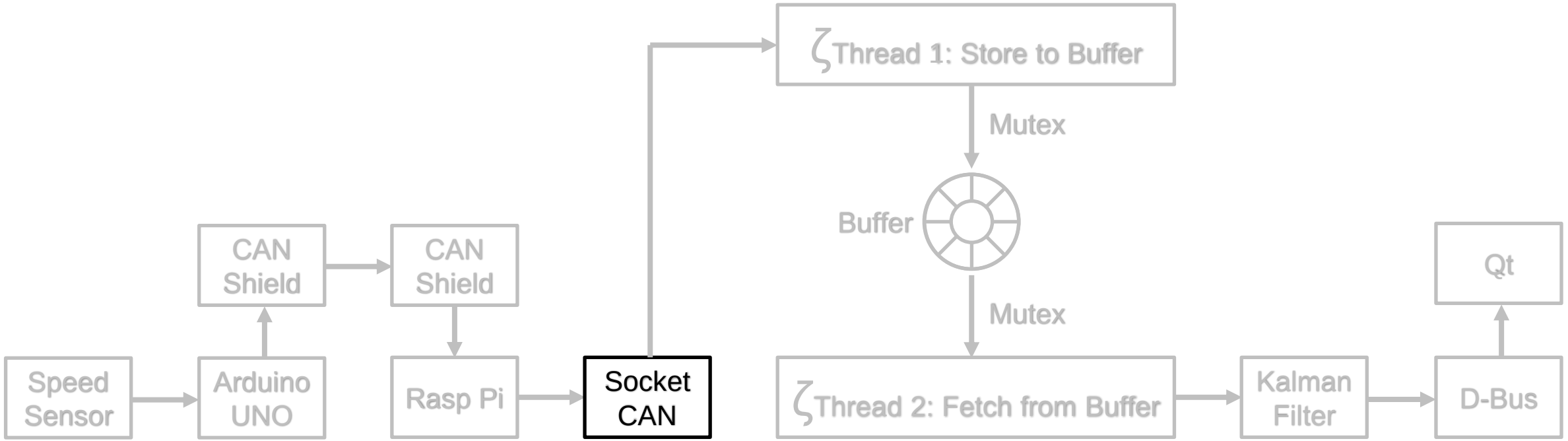
# INDEX

1. Speed Sensor
2. CAN Communication



# INDEX

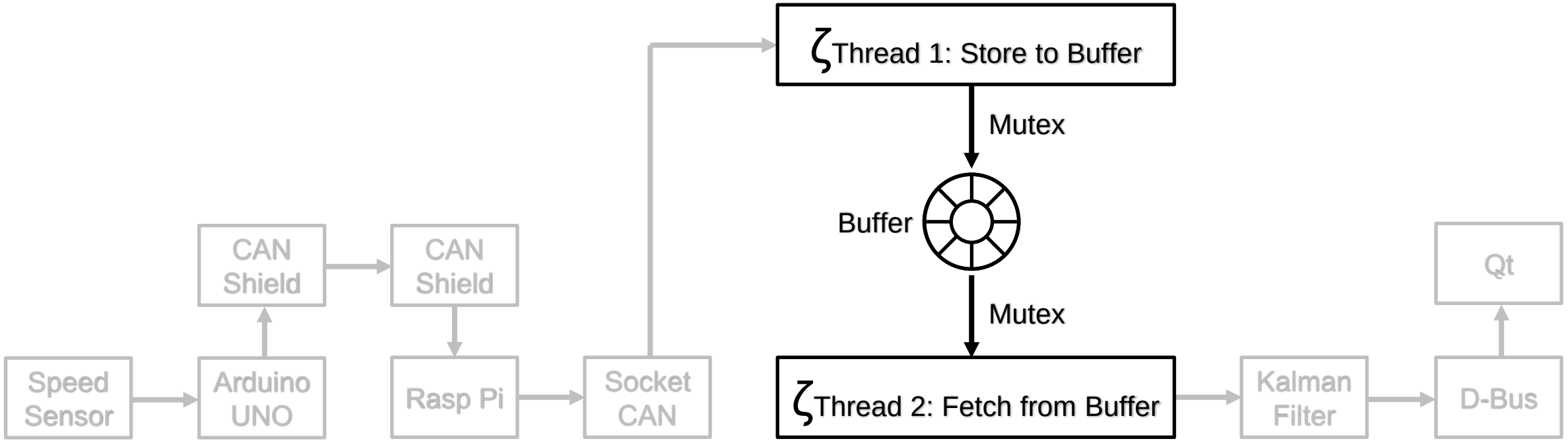




# INDEX

sea|me
software engineering in automotive and mobility ecosystems

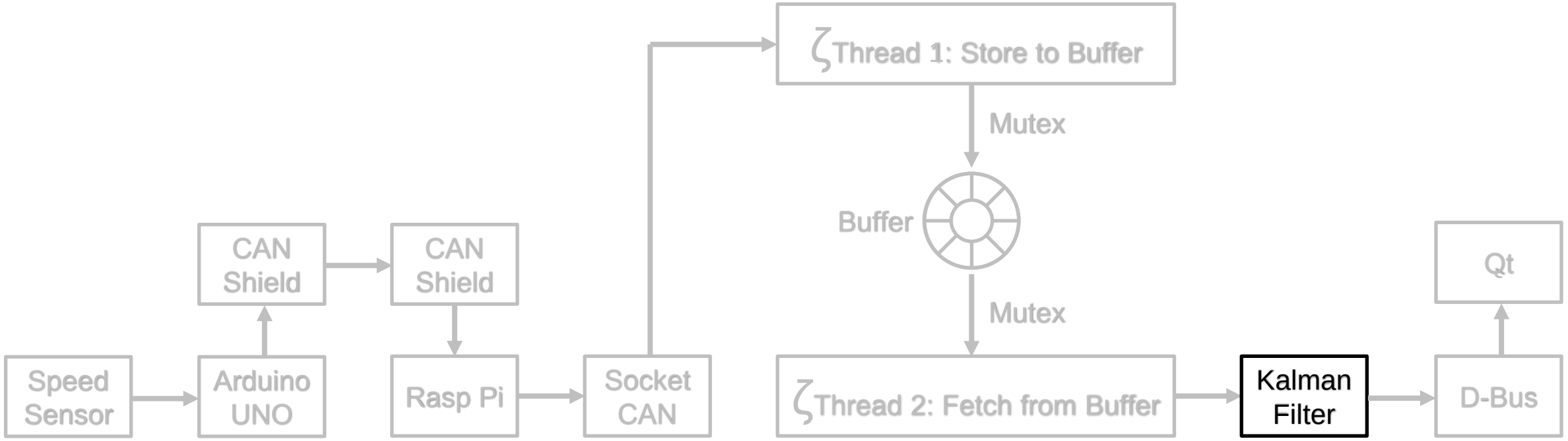
1. Speed Sensor
2. CAN Communication
3. Socket CAN
4. Multi-Thread Synchronization



# INDEX

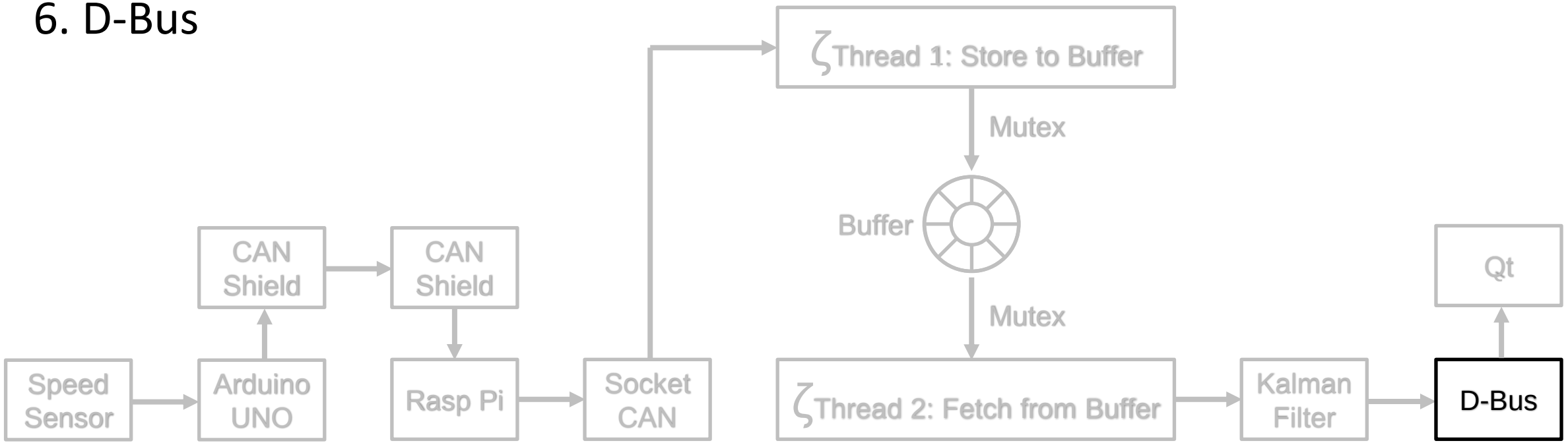
sea|me
software engineering in automotive and mobility ecosystems

1. Speed Sensor
2. CAN Communication
3. Socket CAN
4. Multi-Thread Synchronization
5. Kalman Filter



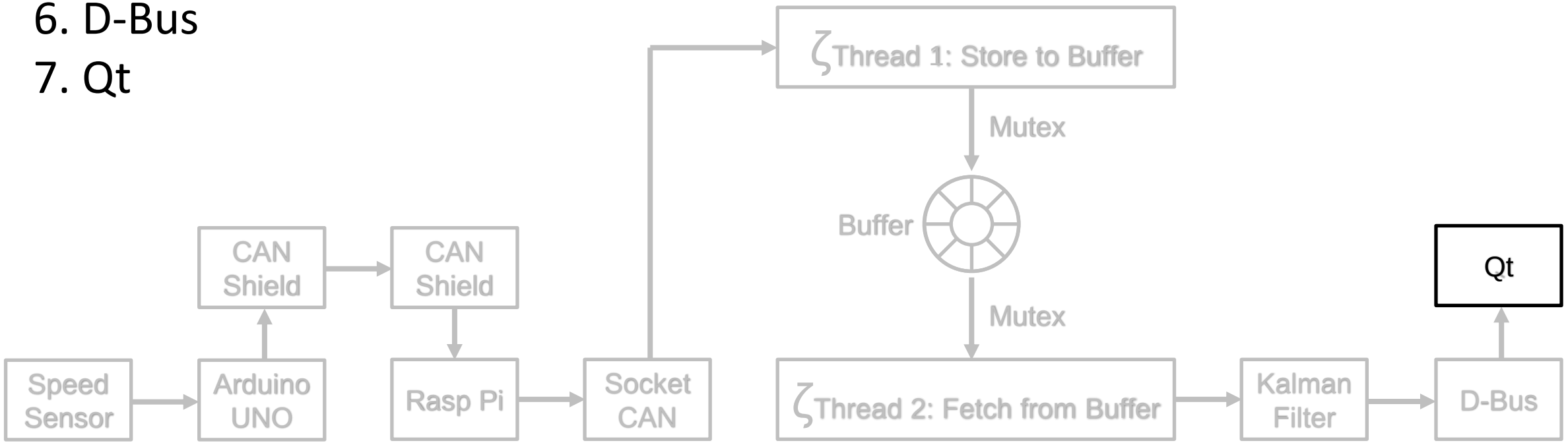
# INDEX

1. Speed Sensor
2. CAN Communication
3. Socket CAN
4. Multi-Thread Synchronization
5. Kalman Filter
6. D-Bus



# INDEX



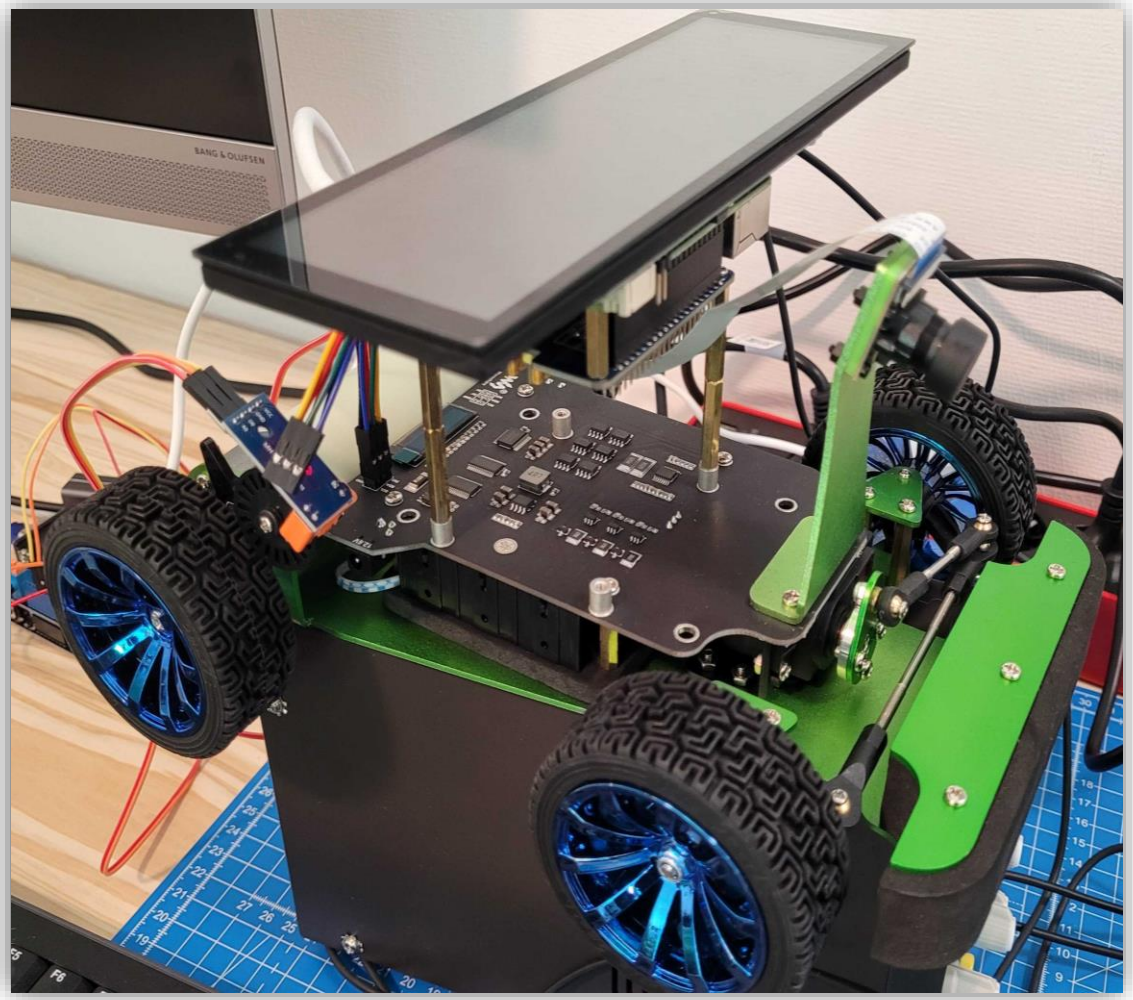


# Hardware & Software Setup

sea|me
software engineering in automotive and mobility ecosystems

## Hardware

- Waveshare Piracer
- Raspberry Pi 4 B 4GB
- 2-CH CAN FD HAT
- 7.9inch DSI LCD
- Arduino UNO
- CAN-Bus Shield
- IR Speed Sensor



## Software

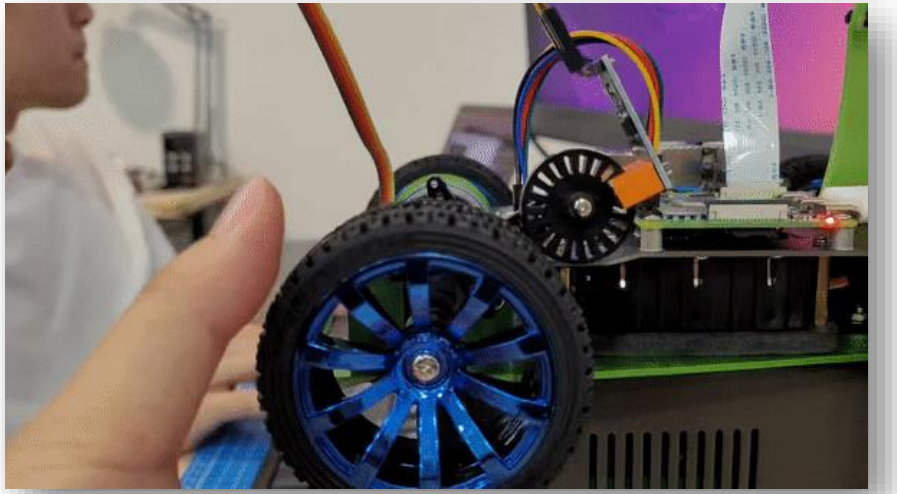
- Rasbian Lite(Server) 64
- Qt 5.15.0
- Qt Creator 4.15.0

# 1. Speed Sensor

sea|me
software engineering in automotive and mobility ecosystems

- Calculate period between 20 pulses and update last measurement time

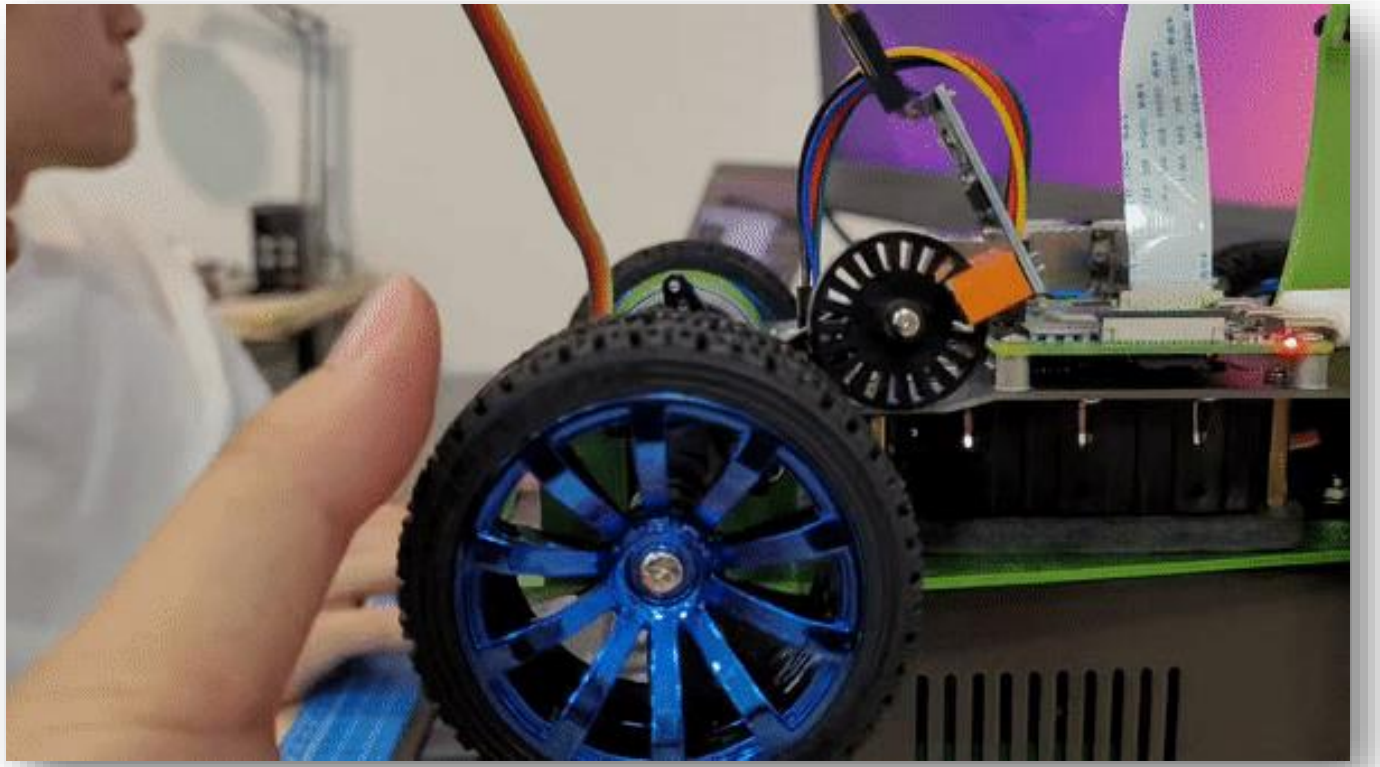
```
void setup()
{
  attachInterrupt(digitalPinToInterrupt(3), Pulse_Event, RISING);
  …
}

void Pulse_Event()
{
  PeriodBetweenPulses = micros() - LastTimeWeMeasured;
  LastTimeWeMeasured = micros();
  …
}
```

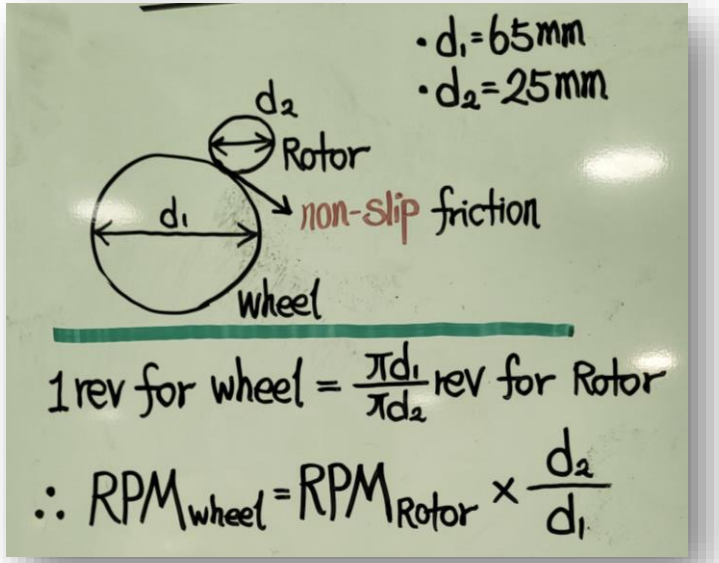


# 1. Speed Sensor

sea|me
software engineering in automotive and mobility ecosystems

- Convert data to RPM & speed of the Piracer



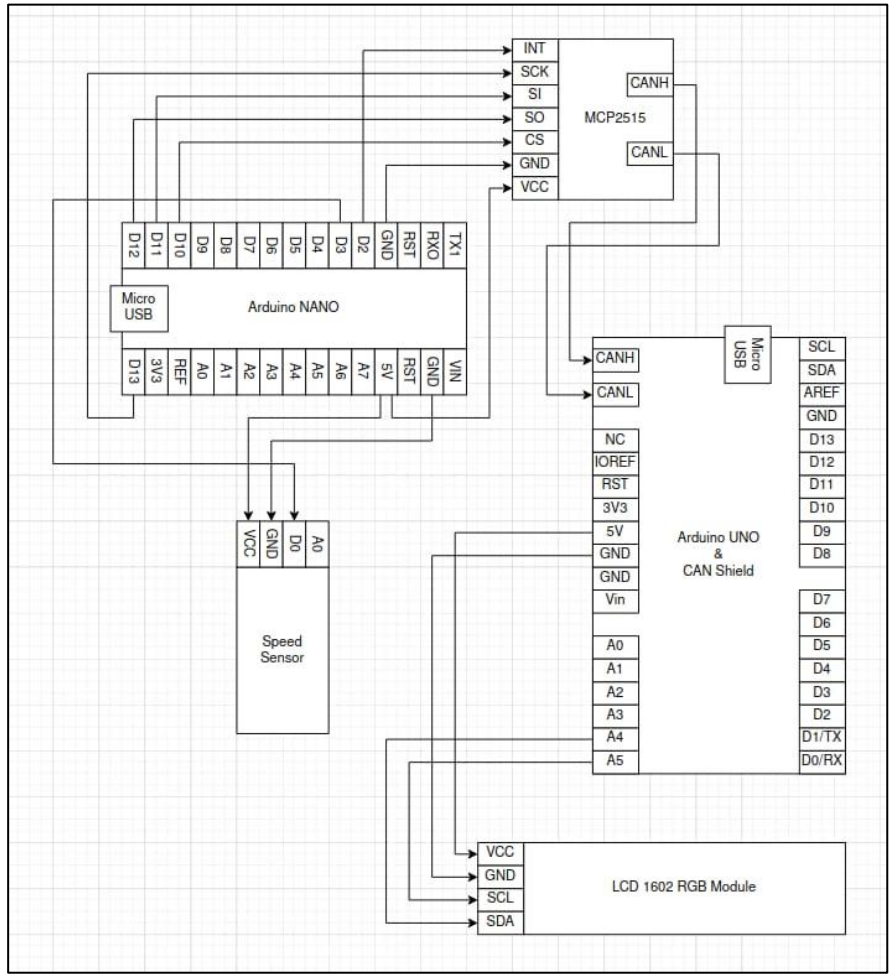



1. $RPM_{wheel} = RPM_{Rotor} / 2.6$
2. $Speed_{Car} = RPM_{Rotor} * 3.4$ [mm/s]

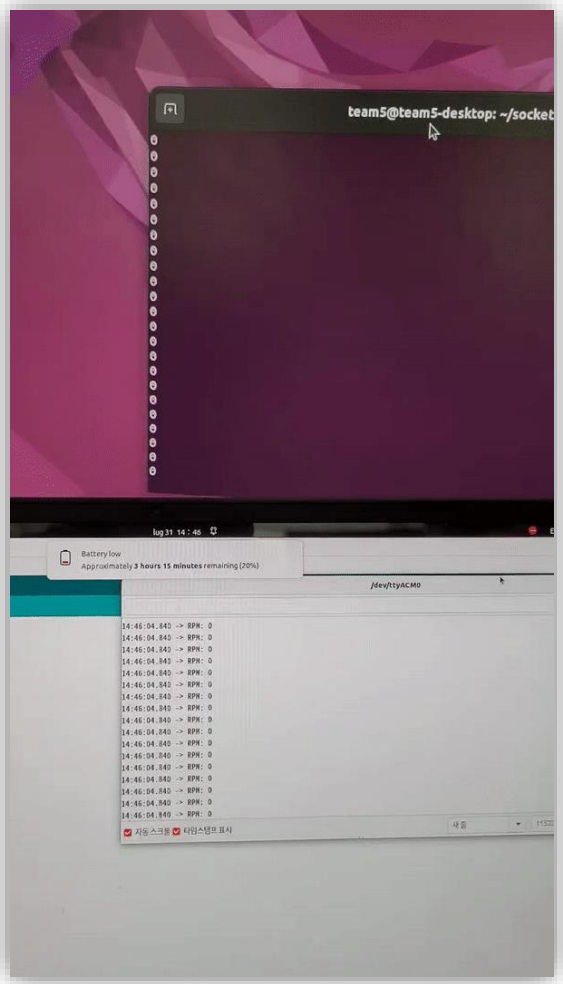
# 2. CAN Communication

sea:me
software engineering in automotive and mobility ecosystems

- Arduino NANO to Arduino UNO




- Arduino UNO to Raspberry Pi




- 500Kbps
- 16MHz

# 2. CAN Communication

sea|me
software engineering in automotive and mobility ecosystems

## Send(Arduino)

```
struct can_frame canMsg1;
…

void loop() {
  …
  canMsg1.can_id  = 0x0F6;
  canMsg1.can_dlc = 2;
  canMsg1.data[0] = (RPM & 0xFF00) >> 8;
  canMsg1.data[1] = (RPM & 0x00FF);

  mcp2515.sendMessage(&canMsg1);
}
```

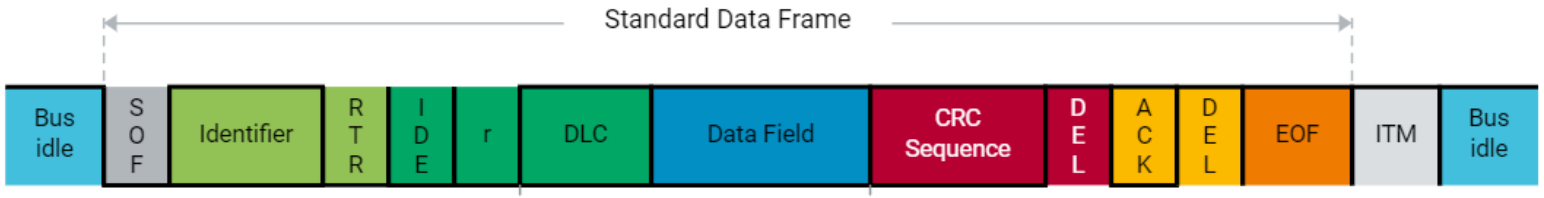
## Receive(C)

```
int open_port(const char *port) {
  …
  struct can_filter rfilter[1];
  …
  rfilter[0].can_id   = 0x0F6;
}

void read_port(uint16_t *speed_sensor_rpm) {
  struct can_frame frame;
  …
  if (frame.can_id == 0x0F6) {
    *speed_sensor_rpm = (frame.data[0] << 8) + frame.data[1];
  }
}
```

# 2. CAN Communication

sea|me
software engineering in automotive and mobility ecosystems

## DLC and Data Field




| DLC_3 | DLC_2 | DLC_1 | DLC_0 | Data Field |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | Byte 0 |
| 0 | 0 | 1 | 0 | Byte 0 Byte 1 |
| 0 | 0 | 1 | 1 | Byte 0 Byte 1 Byte 2 |
| 0 | 1 | 0 | 0 | Byte 0 Byte 1 Byte 2 Byte 3 |
| 0 | 1 | 0 | 1 | Byte 0 Byte 1 Byte 2 Byte 3 Byte 4 |
| 0 | 1 | 1 | 0 | Byte 0 Byte 1 Byte 2 Byte 3 Byte 4 Byte 5 |
| 0 | 1 | 1 | 1 | Byte 0 Byte 1 Byte 2 Byte 3 Byte 4 Byte 5 Byte 6 |
| 1 | 0 | 0 | 0 | Byte 0 Byte 1 Byte 2 Byte 3 Byte 4 Byte 5 Byte 6 Byte 7 |

**CAN Frame Structure**

# 3. Socket CAN

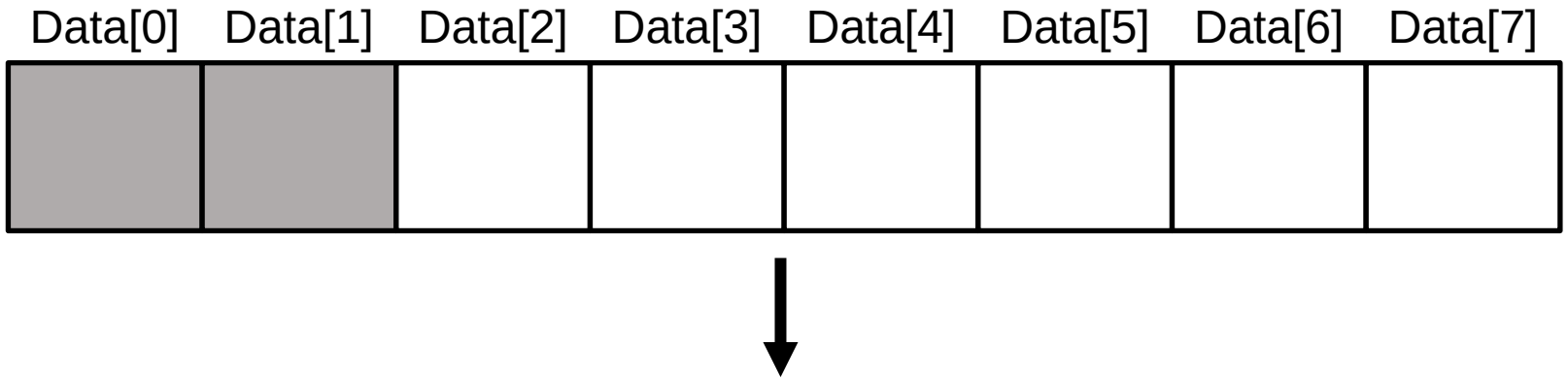
sea|me
software engineering in automotive and mobility ecosystems

let RPM = 300 = $100101100_{(2)}$

(**RPM** & 0xFF00) >> 8

(**RPM** & 0x00FF)

00000001 00101100




**RPM** = (frame.data[0] << 8) + frame.data[1]

# 3. Socket CAN

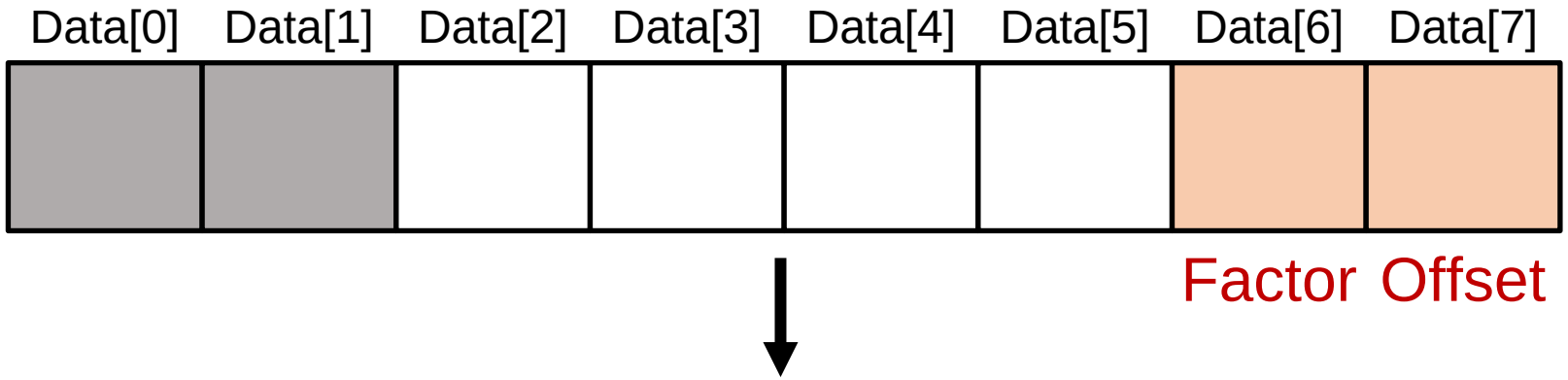
sea|me
software engineering in automotive and mobility ecosystems

let RPM = 300 = $100101100_{(2)}$

(**RPM** & 0xFF00) >> 8

(**RPM** & 0x00FF)

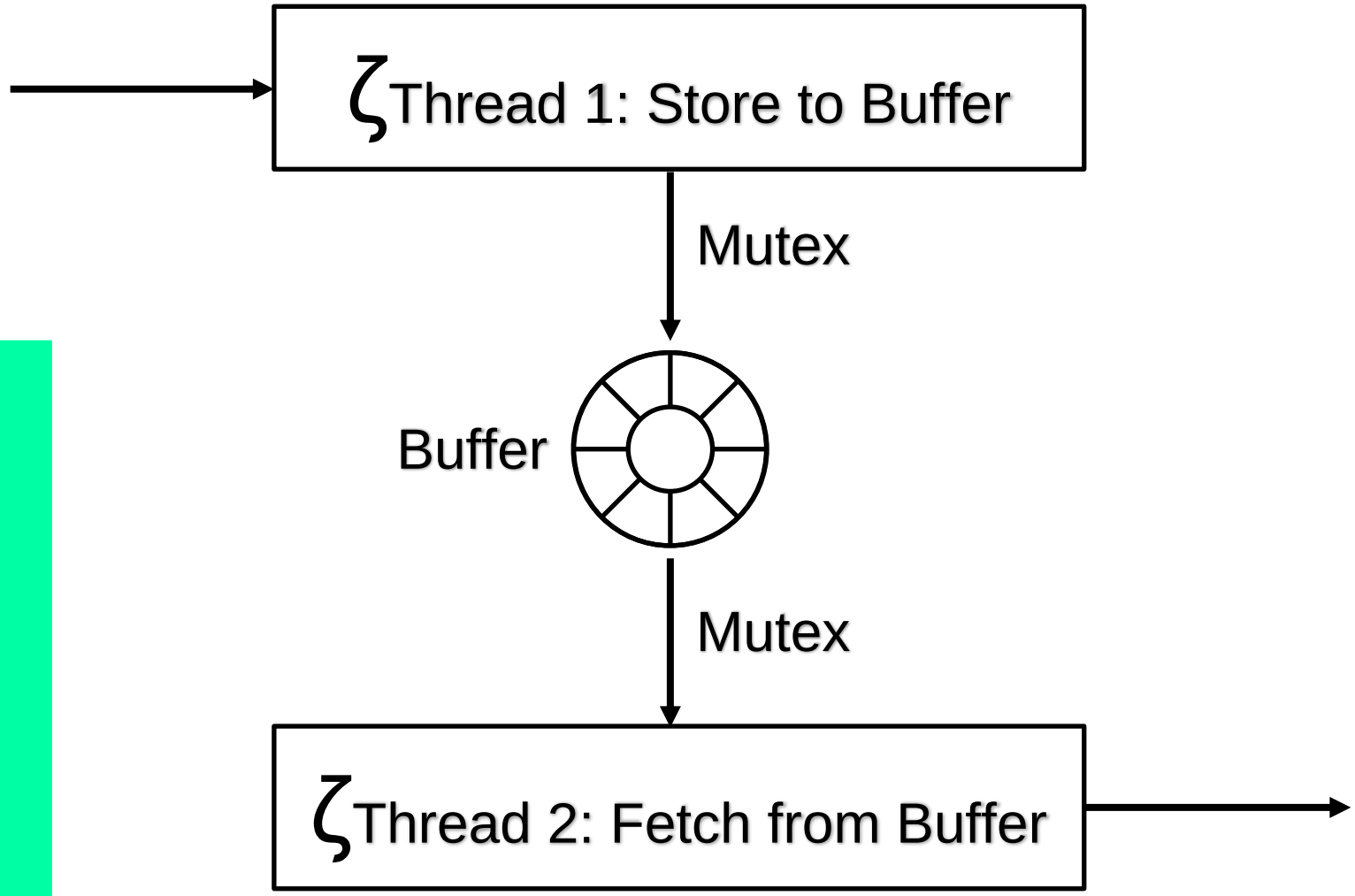
00000001 00101100



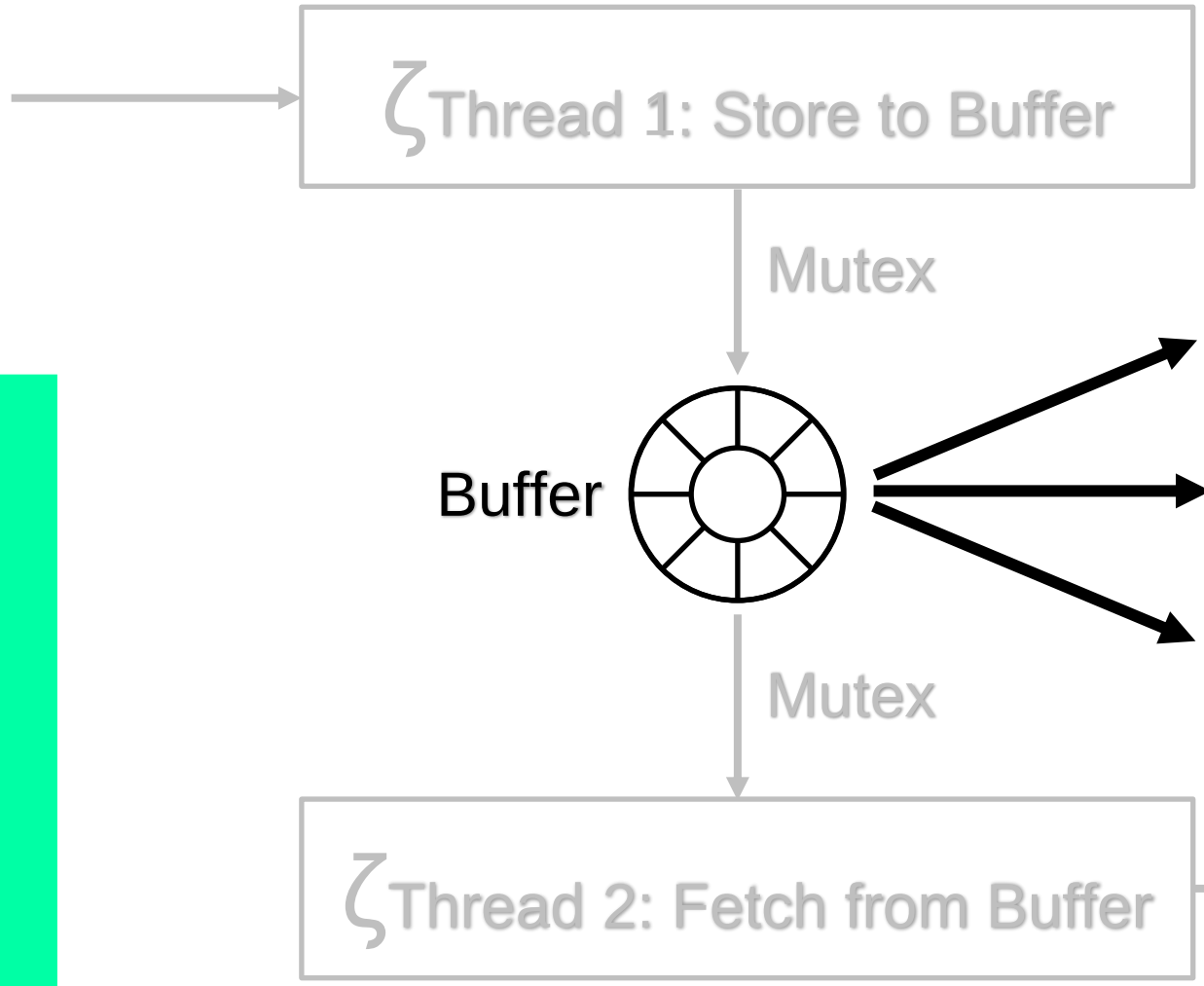

**RPM** = (frame.data[0] << 8) + frame.data[1]

# 4. Multi-Thread Synchronization

sea|me
software engineering in automotive and mobility ecosystems

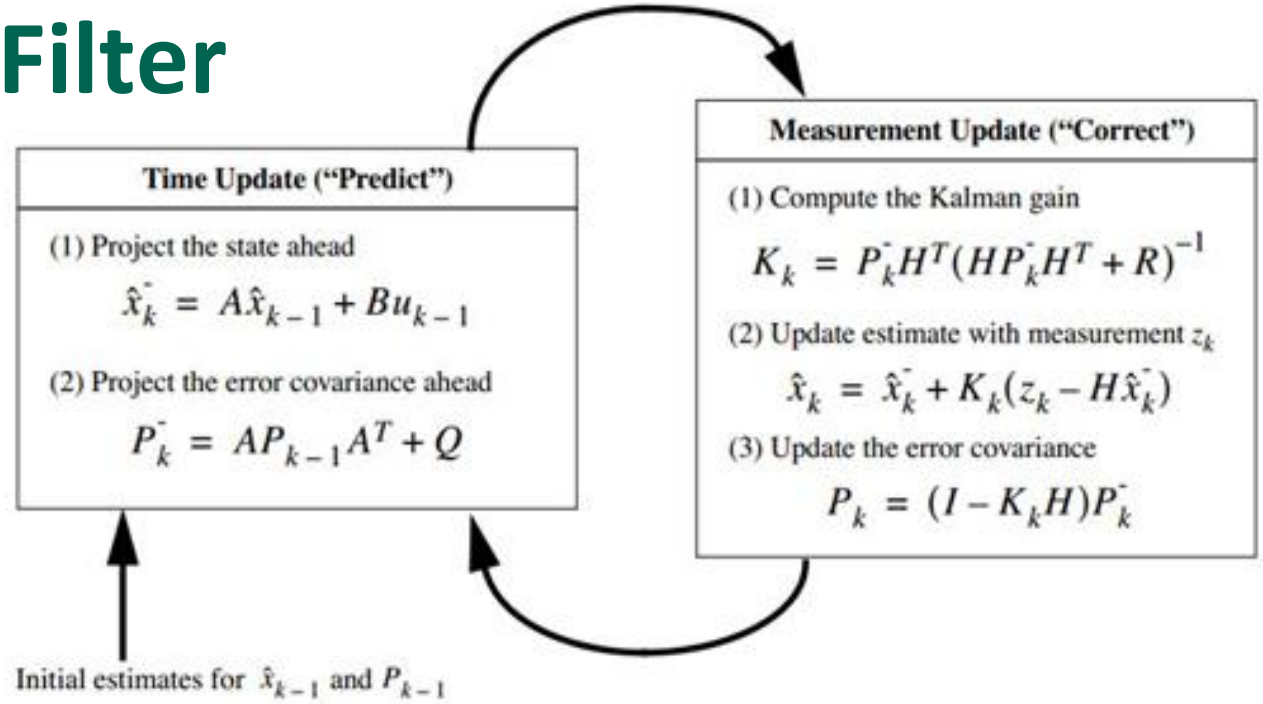


# 4. Multi-Thread Synchronization




- Optimizing data processing speed
- Transmission Overhead
- Resource Usage

# 5. Kalman Filter




A, H : State Space Equation

$$Ax$$

$$x = \begin{bmatrix} Position \\ Speed \end{bmatrix} \qquad A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

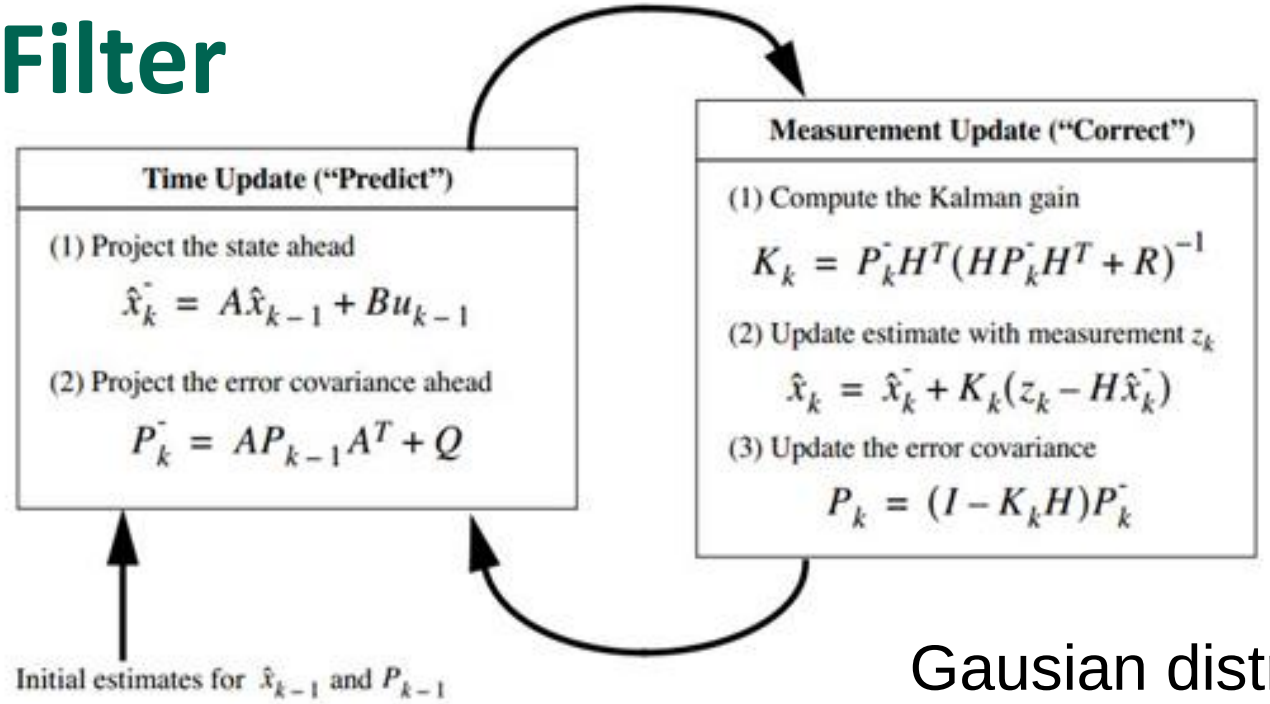
$$current\_position = past\_position + \Delta t * past\_speed$$
$$current\_speed = past\_speed$$

$$x = \begin{bmatrix} Velocity \\ Acceleration \\ \Delta^2 Acceleration \end{bmatrix} \qquad A = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

→ Numerical Analysis

# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems




Q: System mode1 noise

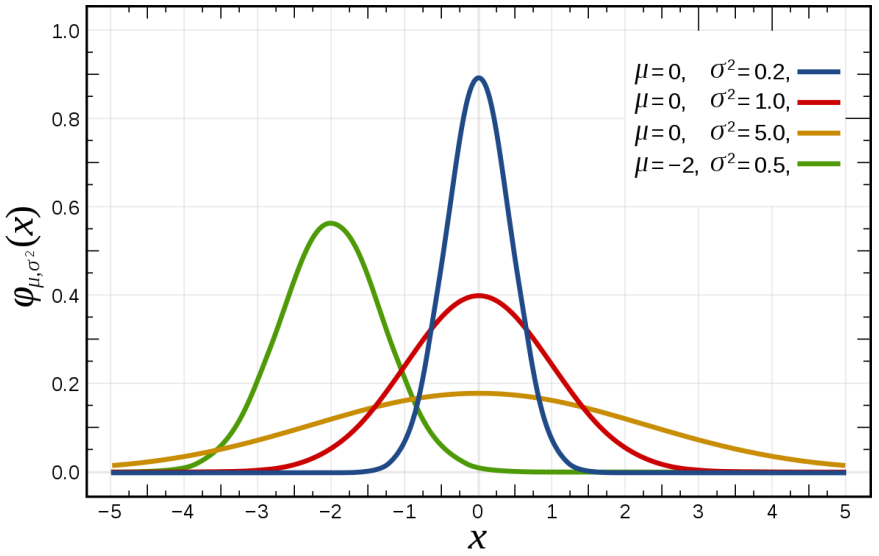
R: Sensor mode1 noise

P: Relative formula

Z: Actual data

K: Kalman gain

Gausian distribution



# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

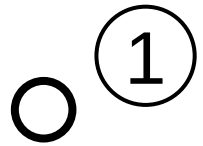
● : Filtered data

# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

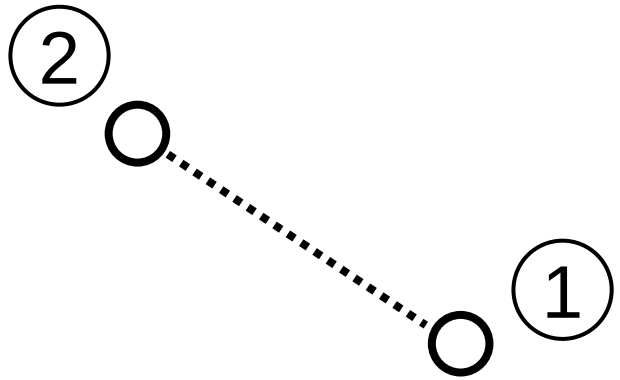


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

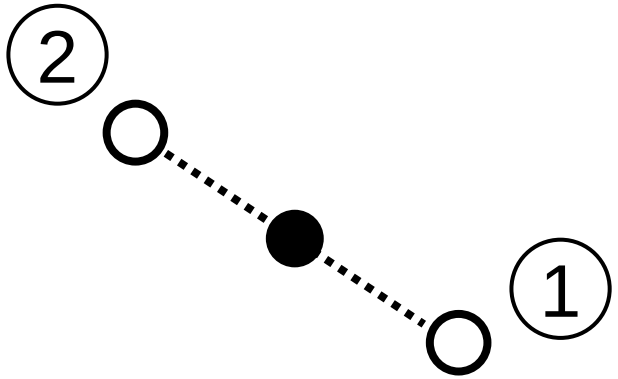


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

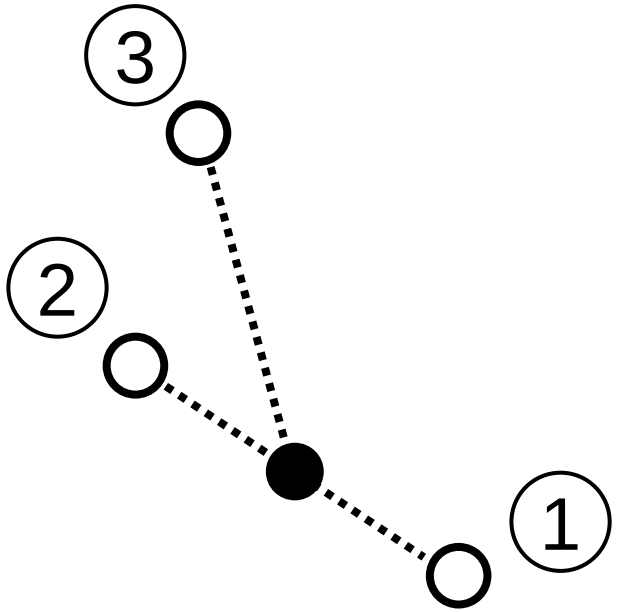


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

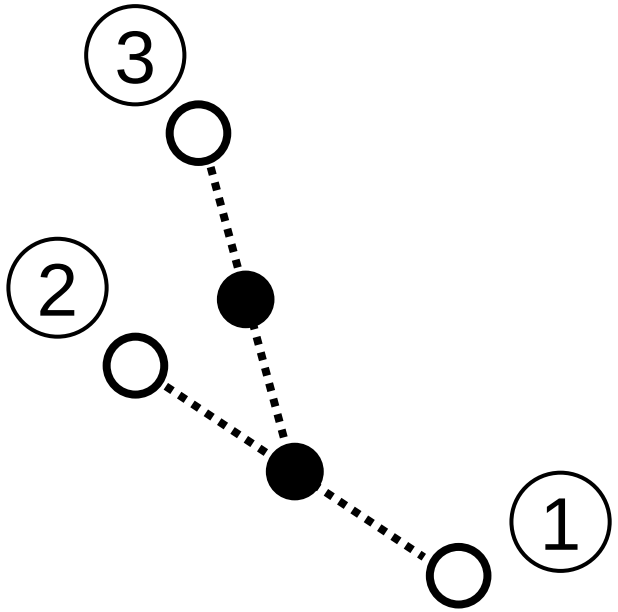


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

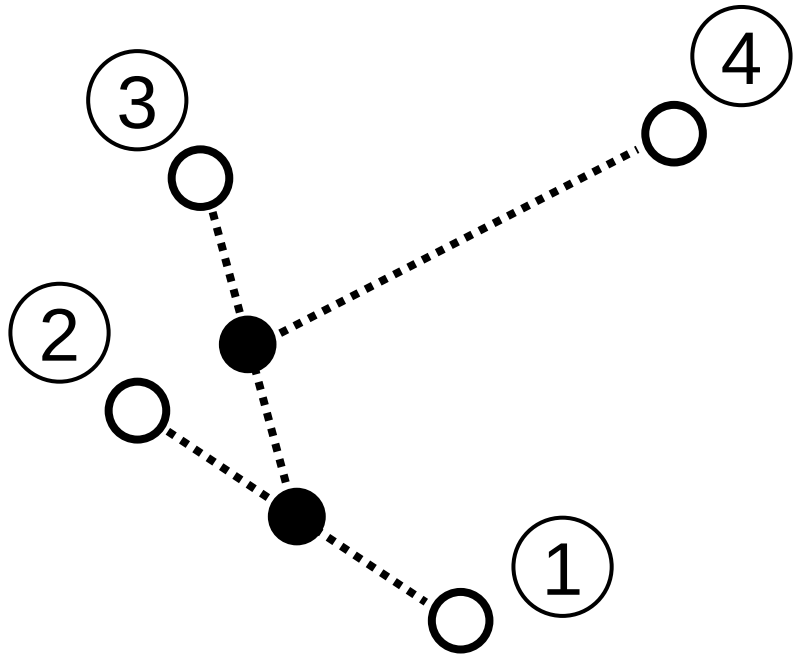


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

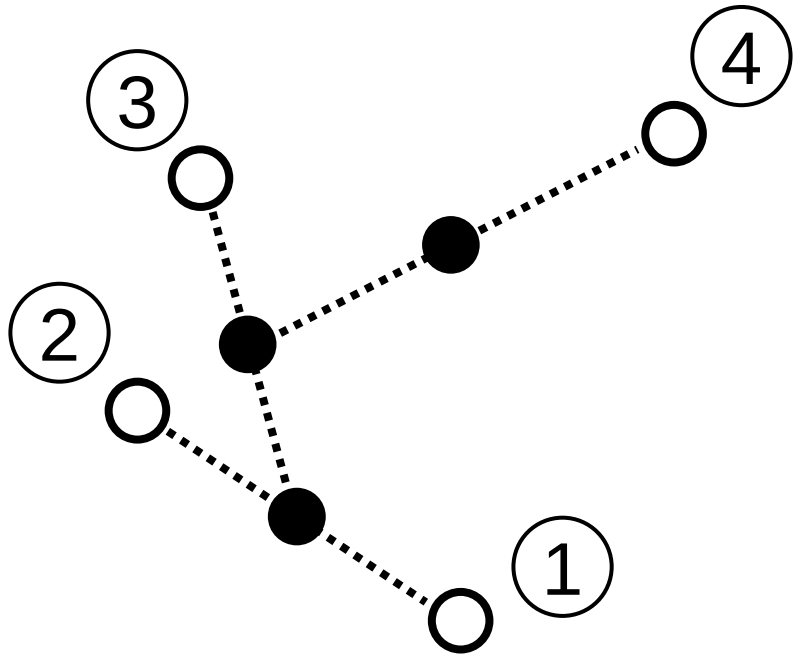


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

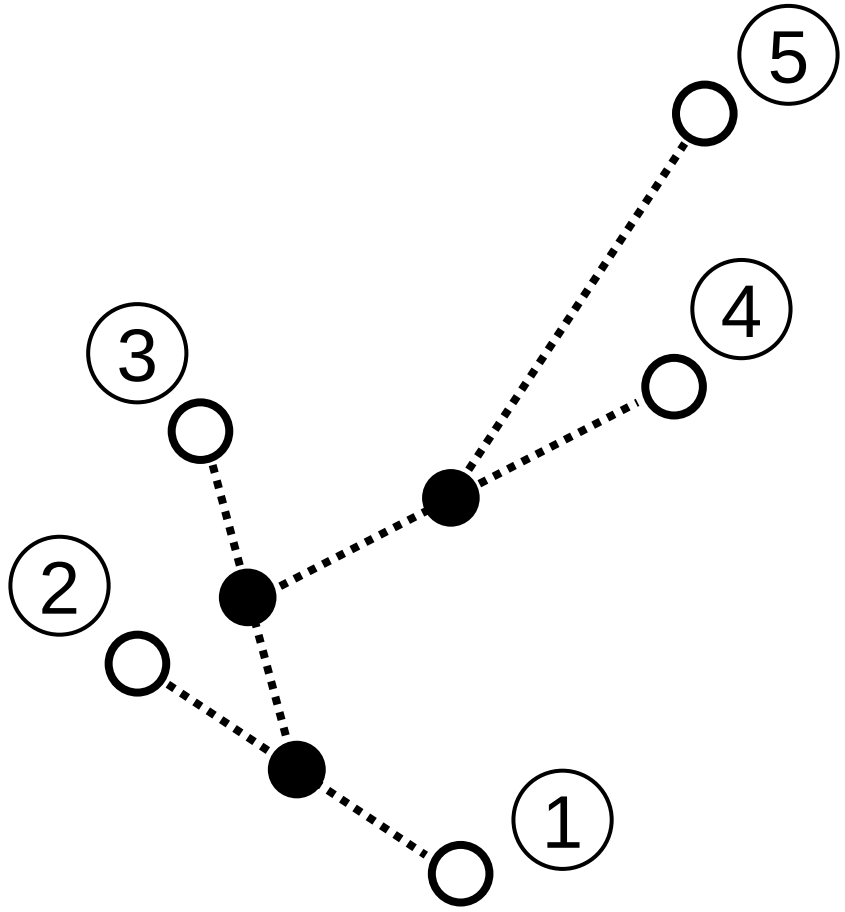


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

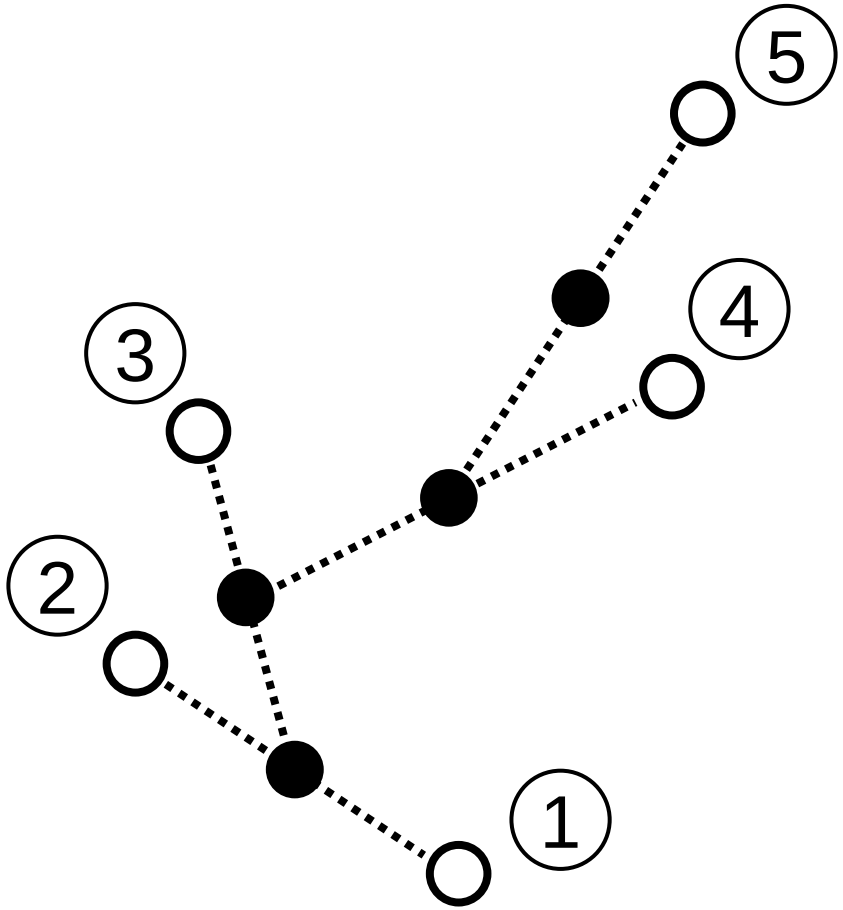


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

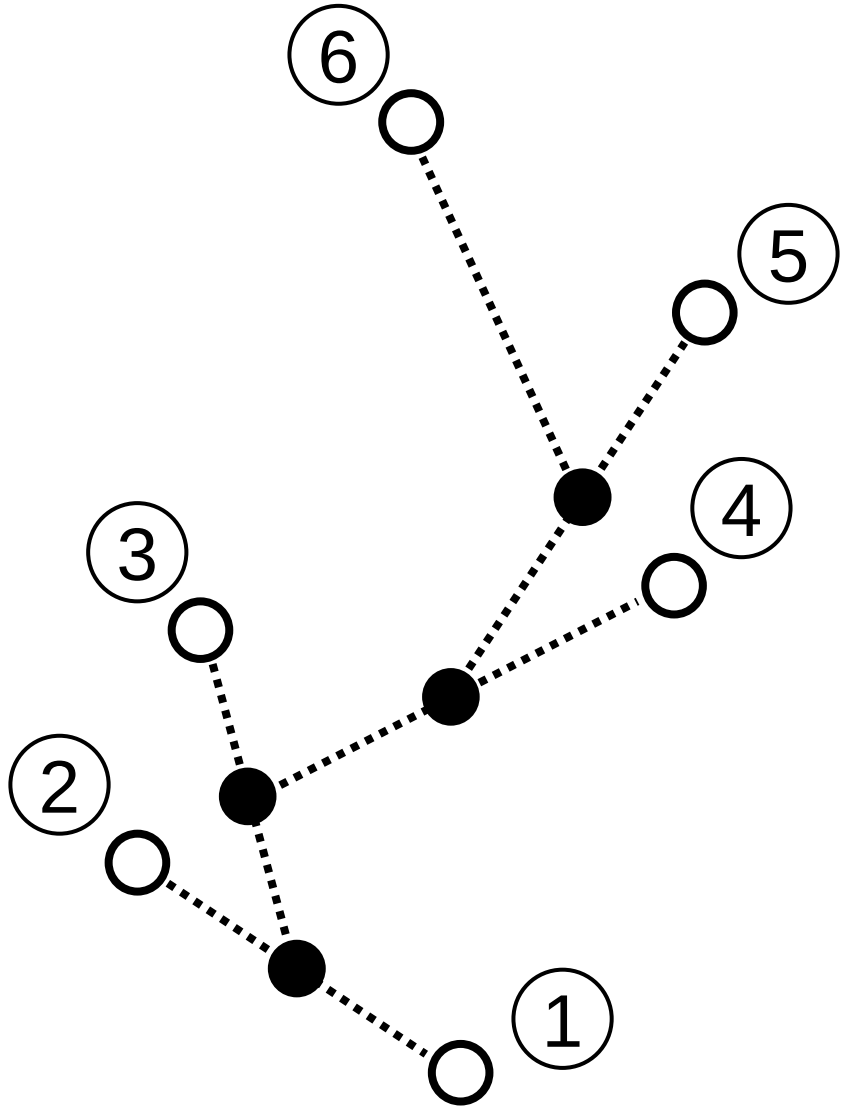


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

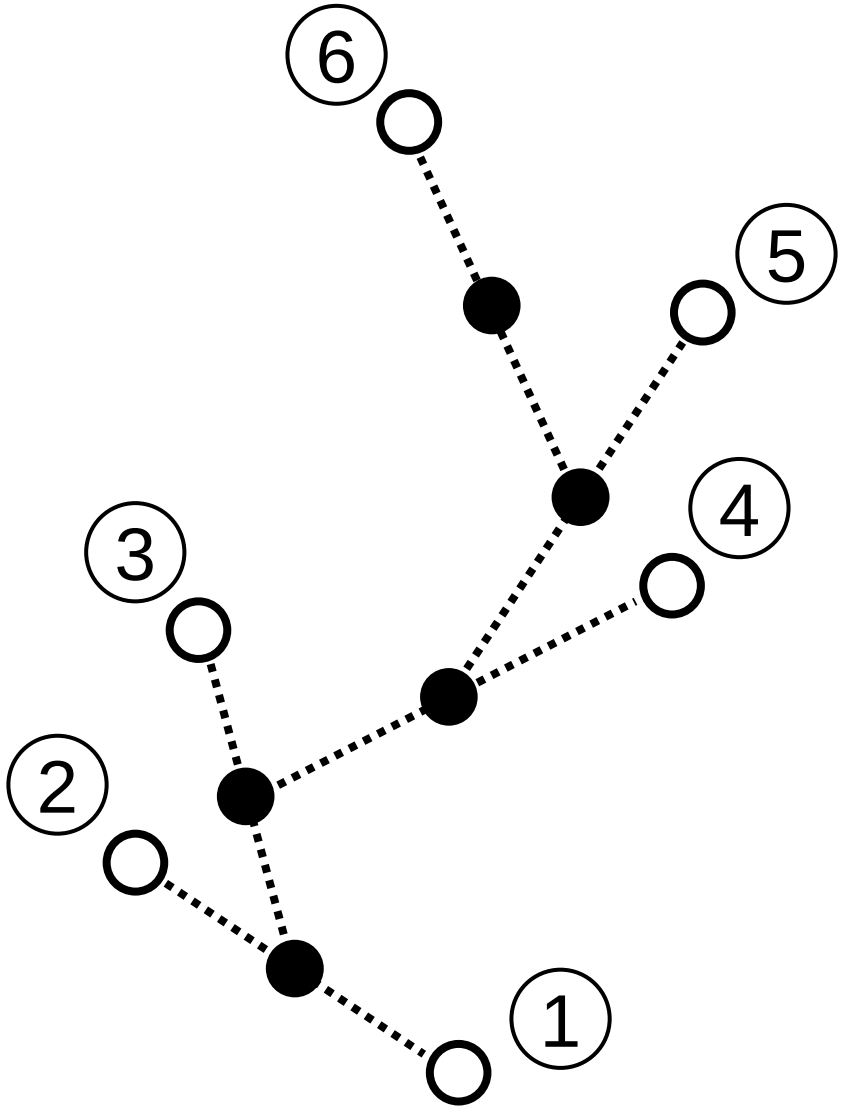
● : Filtered data



# 5. Kalman Filter

○ : Raw data

● : Filtered data



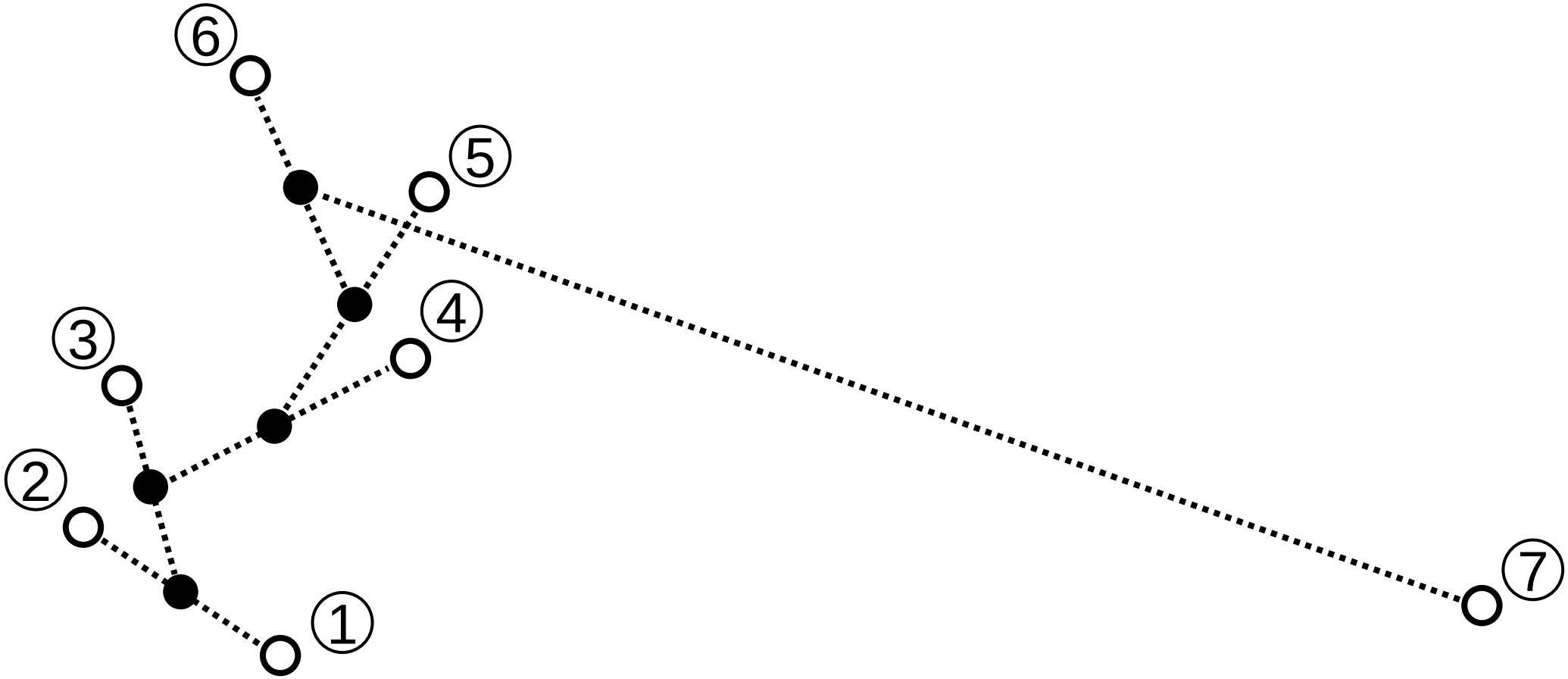

sea|me
software engineering in automotive and mobility ecosystems

# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

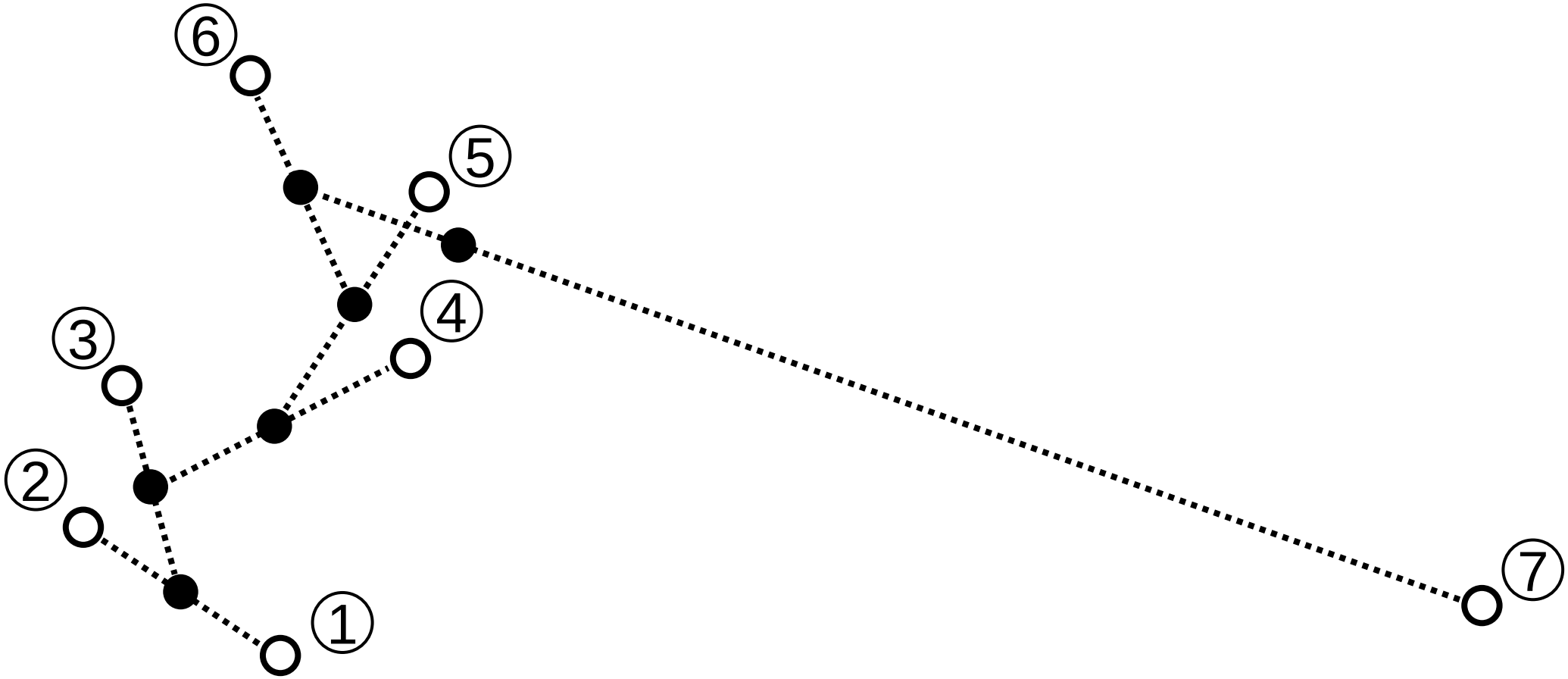


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

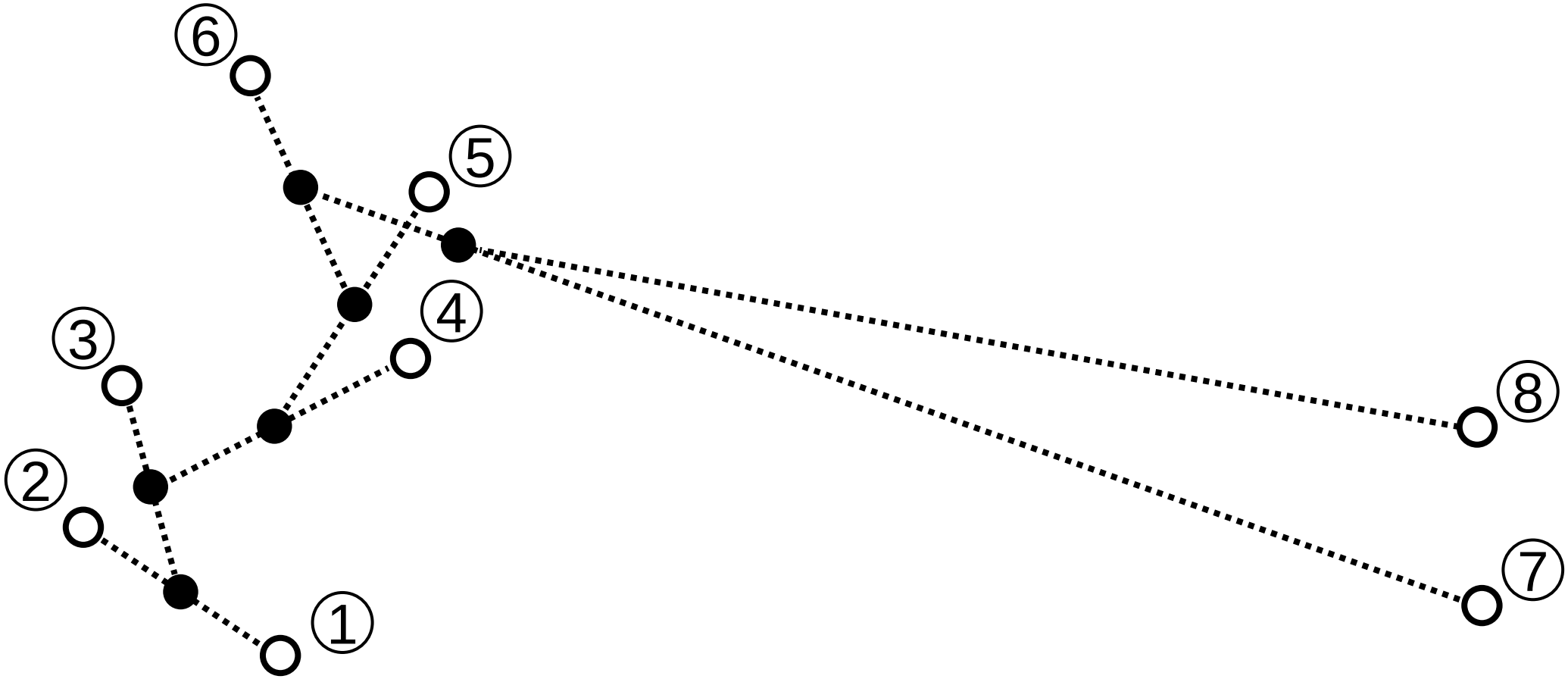


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

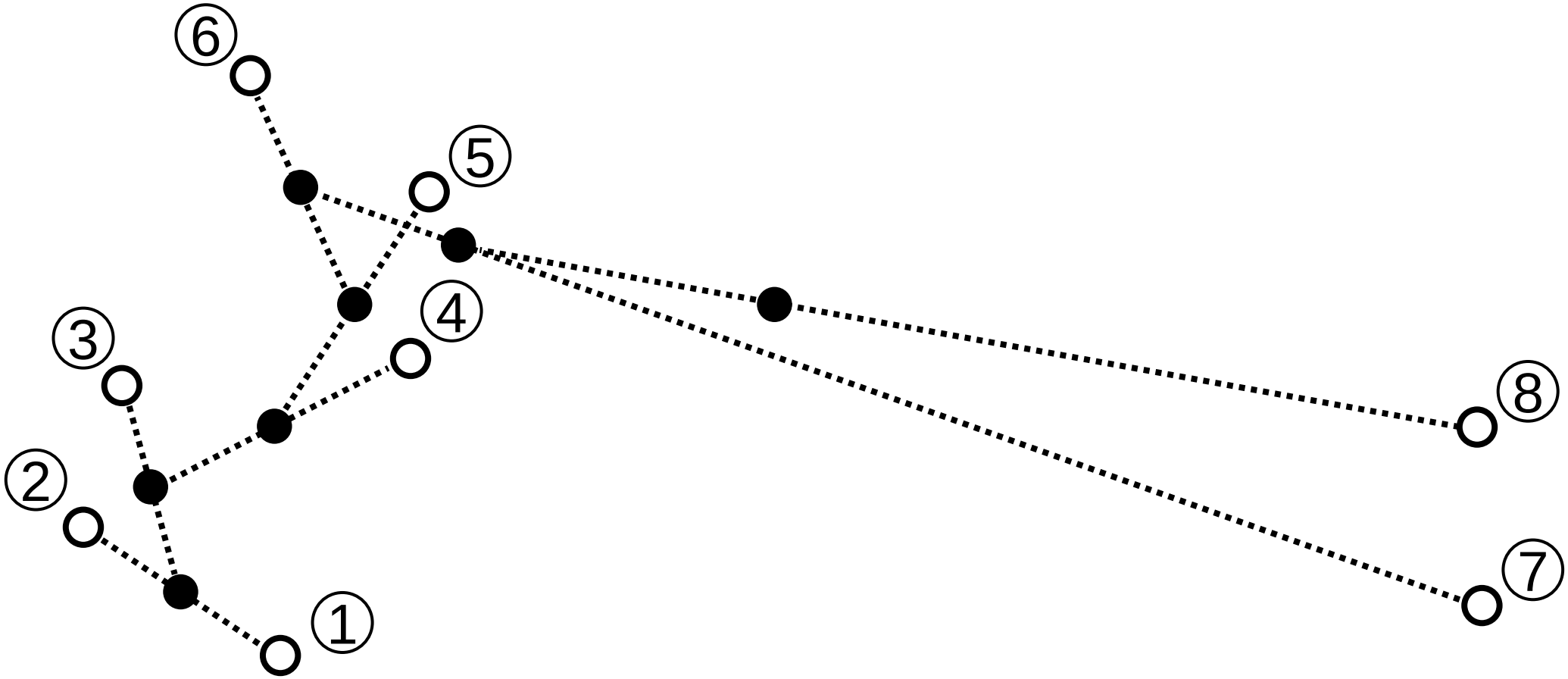


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

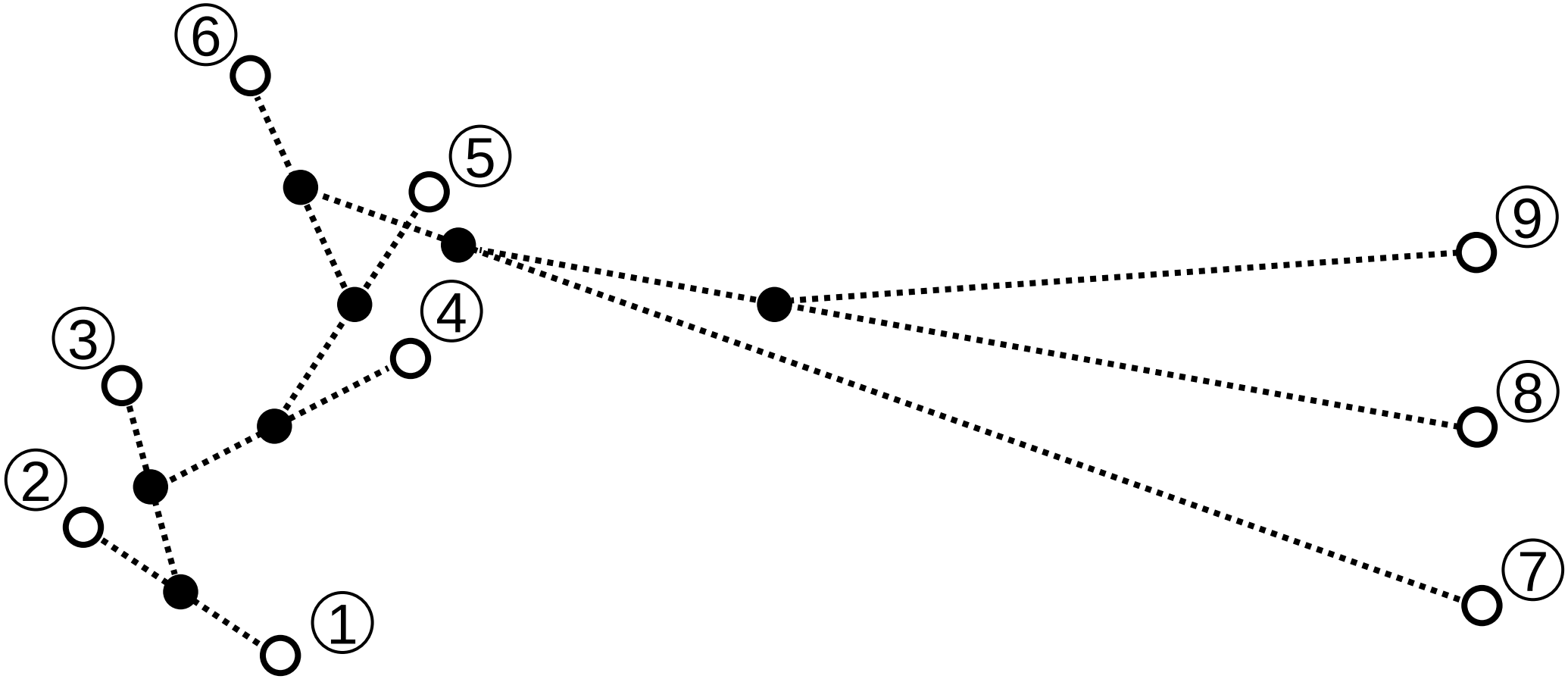


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

○ : Raw data

● : Filtered data

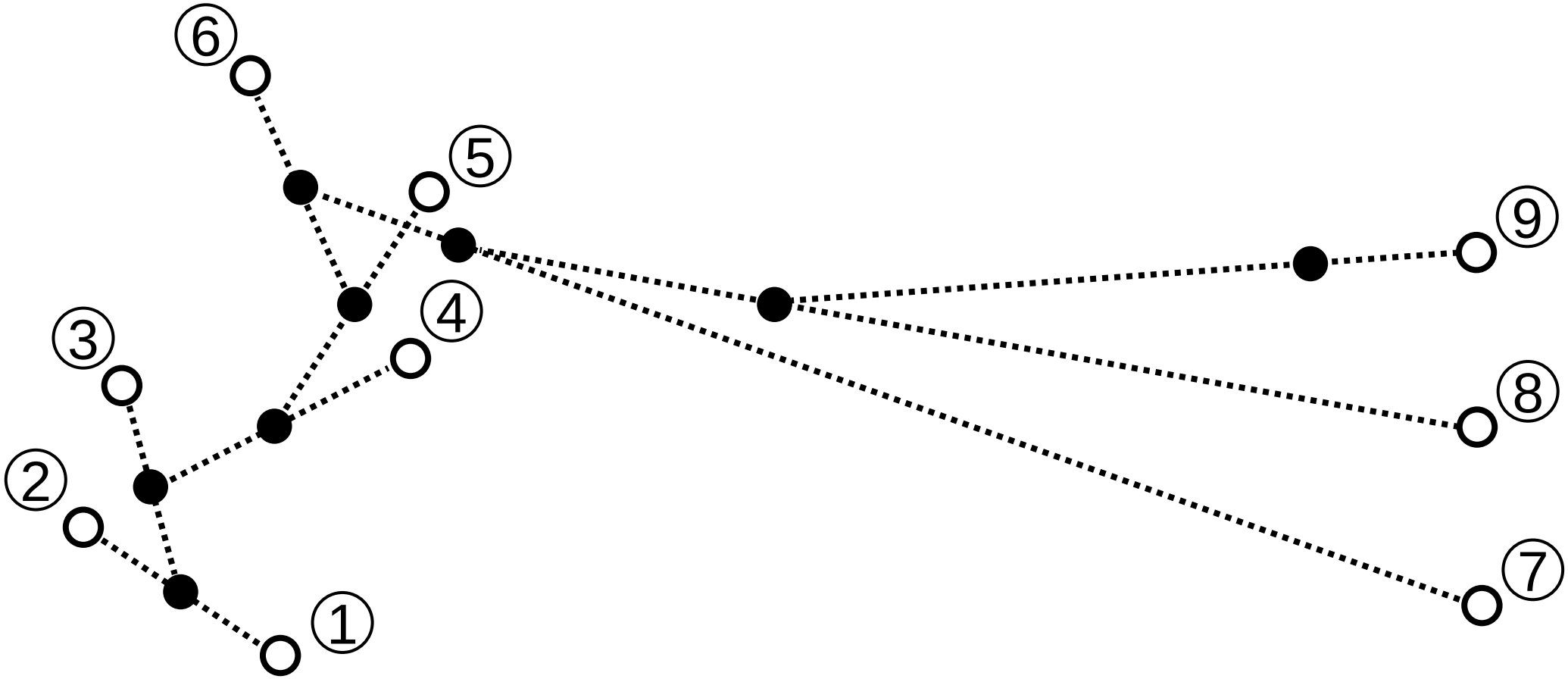


# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems

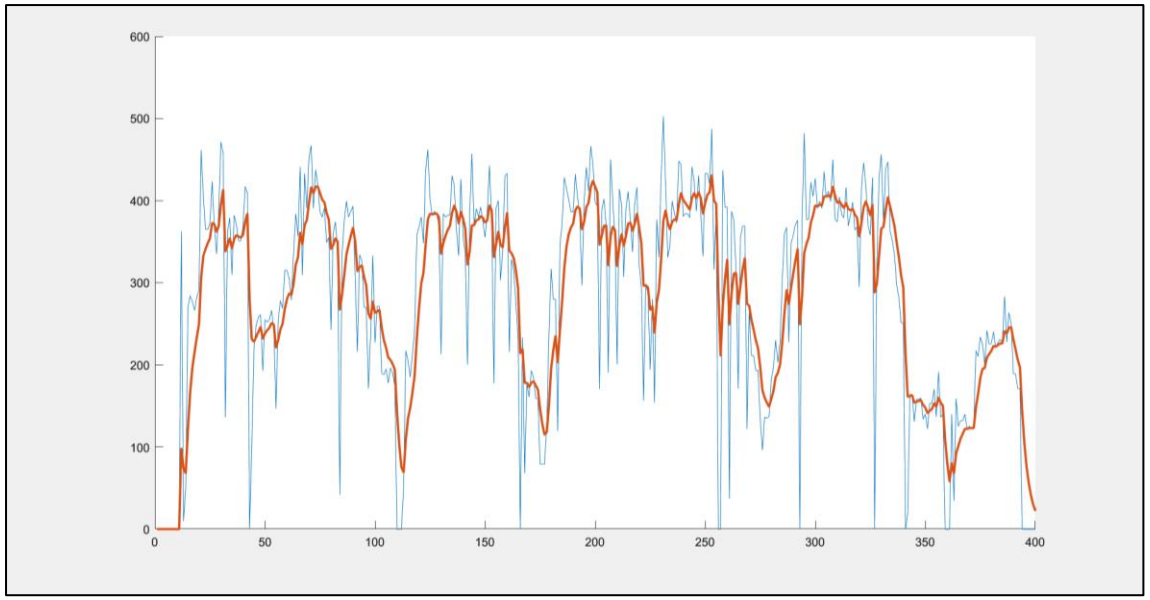
○ : Raw data

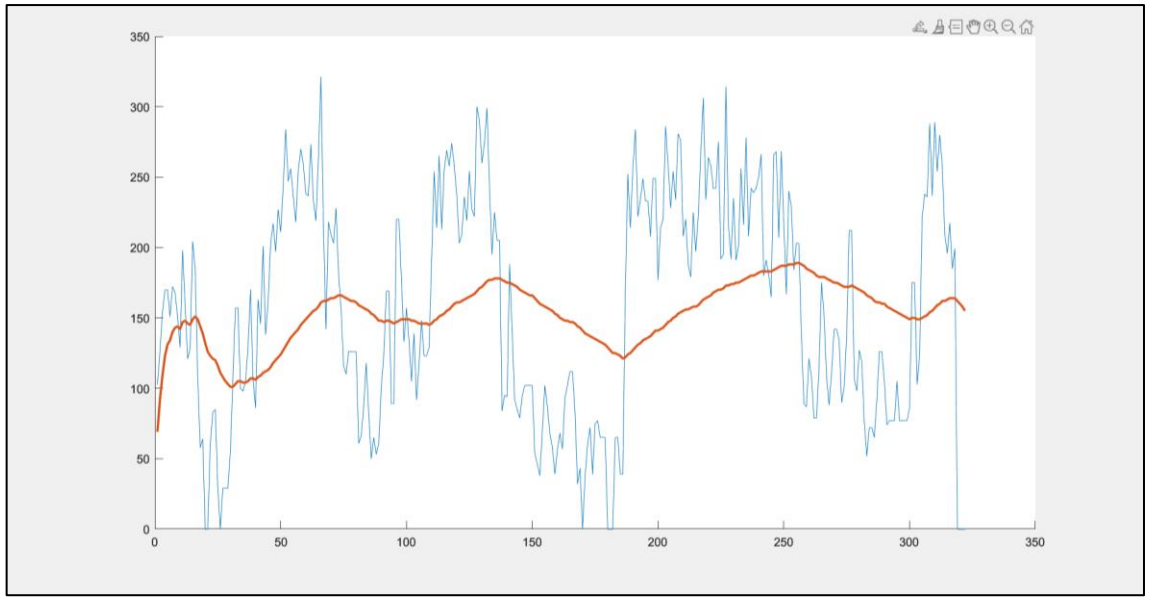
● : Filtered data



# 5. Kalman Filter

sea|me
software engineering in automotive and mobility ecosystems




Different Q, R calibration

# 6. D-Bus

sea|me
software engineering in automotive and mobility ecosystems

```
// Create and initialize the speed message for dbus
speed_msg = dbus_message_new_method_call(SERVER_BUS_NAME, SERVER_OBJECT_PATH_NAME, INTERFACE_NAME, "setSpeed");
if (speed_msg == NULL)
{
    fprintf(stderr, "Speed Message Null\n");
    exit(1);
}
dbus_message_iter_init_append(speed_msg, &speed_args);
if (!dbus_message_iter_append_basic(&speed_args, DBUS_TYPE_UINT16, &speed_value))
{
    fprintf(stderr, "Out Of Memory for setSpeed!\n");
    exit(1);
}

// Send the speed message and wait for a reply
if (!dbus_connection_send_with_reply(conn, speed_msg, &speed_pending, -1))
{
    fprintf(stderr, "Out Of Memory for setSpeed!\n");
    exit(1);
}

dbus_connection_flush(conn); // Send all CANbuffered data
dbus_message_unref(speed_msg); // Unreference the speed message
```

# 6. D-Bus

## DBusPendingCall

sea|me
software engineering in automotive and mobility ecosystems

synchronous → dbus_pending_call_block

asynchronous → dbus_pending_call_set_notify

```
// Block until we receive a reply for the speed message
dbus_pending_call_block(speed_pending);
speed_reply = dbus_pending_call_steal_reply(speed_pending);
if (speed_reply == NULL)
{
    fprintf(stderr, "Speed Reply Null\n");
    exit(1);
}
char *speed_reply_msg;
// Extract the reply message string and print it
if (dbus_message_get_args(speed_reply, &dbus_error, DBUS_TYPE_STRING, &speed_reply_msg, DBUS_TYPE_INVALID))
{
    printf("setSpeed Reply: %s\n", speed_reply_msg);
}
dbus_message_unref(speed_reply);
dbus_pending_call_unref(speed_pending);
```

# 7. Qt

sea|me
software engineering in automotive and mobility ecosystems

## Send(C, Python)

QString setSpeed(quint16 speed);

QString setBattery(qreal battery);

QString setGear(quint8 gear);

QString setRpm(quint16 rpm)

## Receive(Qt)

Q_INVOKABLE quint16 getSpeed();

Q_INVOKABLE qreal getBattery();

Q_INVOKABLE quint8 getGear();

Q_INVOKABLE quint16 getRpm();

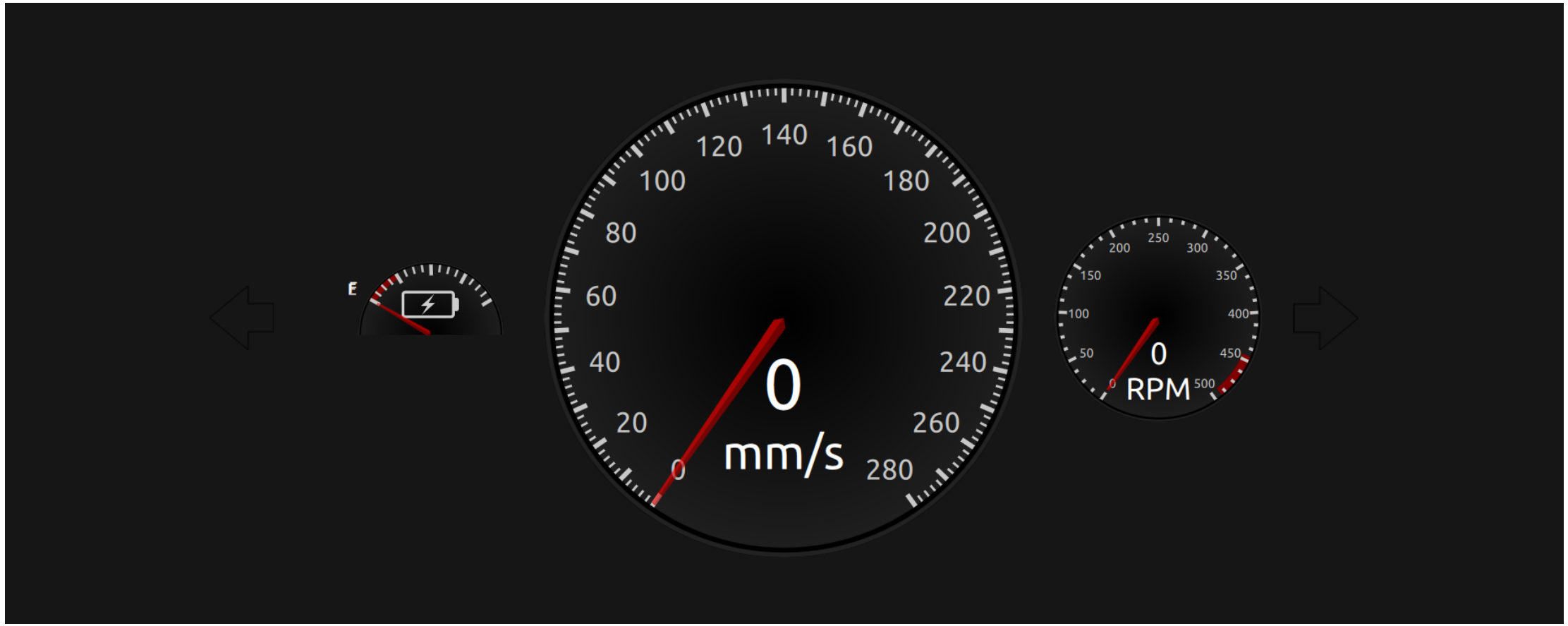
# 7. Qt

sea|me
software engineering in automotive and mobility ecosystems

```
function run_ui() {
    valueSource.speed = carinfo.getSpeed()
    valueSource.battery = carinfo.getBattery()
    valueSource.rpm = carinfo.getRpm()
    …
}

Timer {
    interval: 250; running: true; repeat: true
    onTriggered: valueSource.run_ui()
}
```

# 7. Qt

sea|me
software engineering in automotive and mobility ecosystems

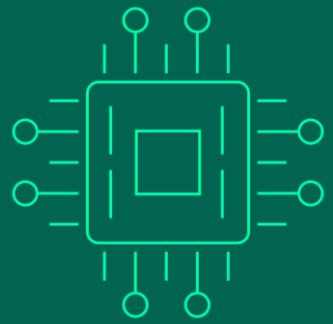


sea|me

software engineering in automotive
and mobility ecosystems

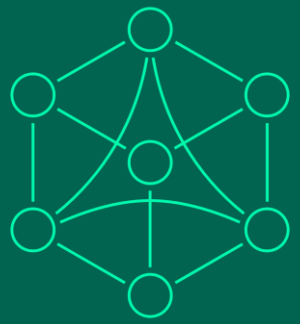
# Thank you!

# Q&A



DongHwan Seo

Mounika Vinjarapu



Sujong Ha