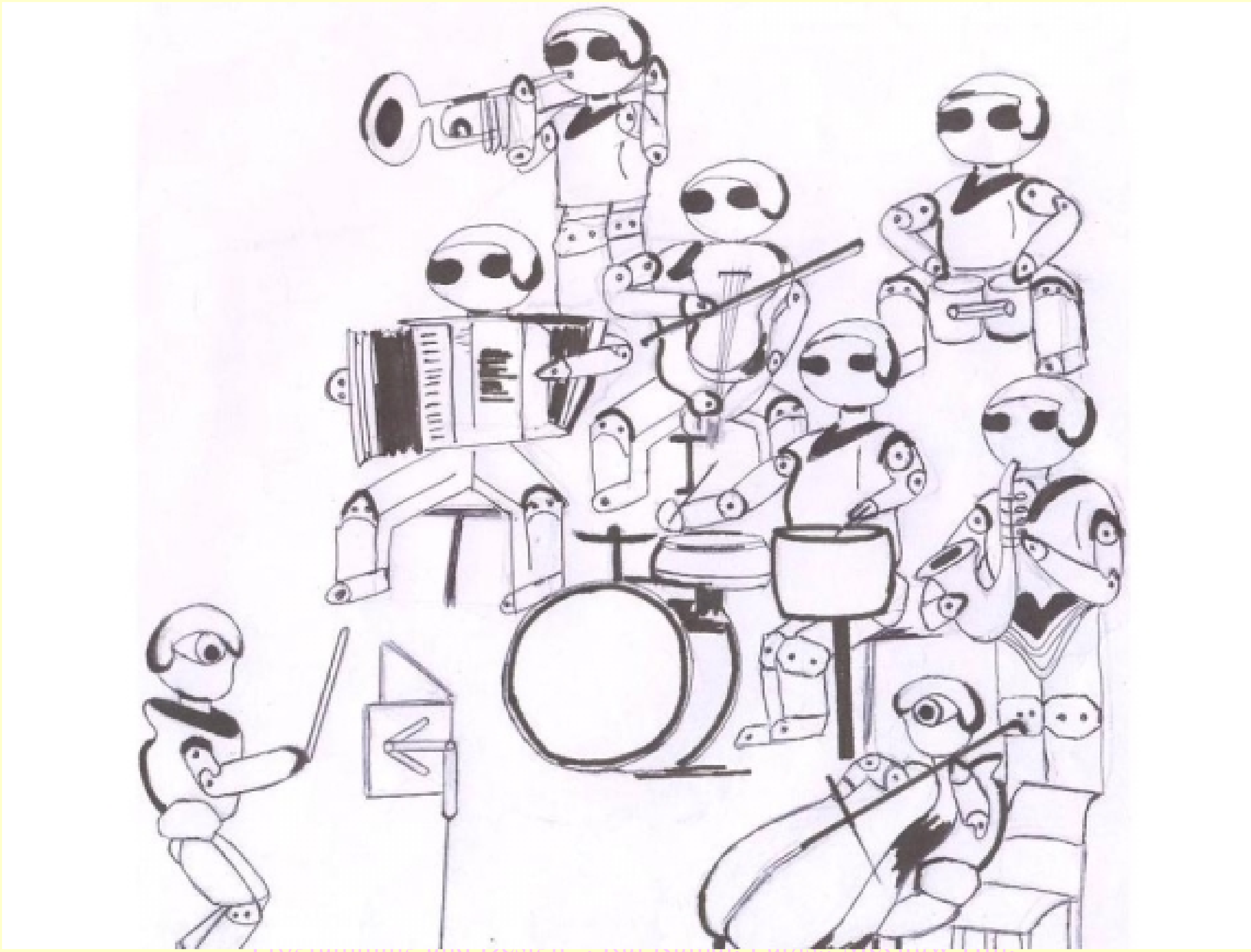
Design Examples and Case Studies of Program Modeling and Programming with RTOS-2:

Lesson-1

Case Study of Inter-Robot Communication in a Robot Orchestra

1. Example of a practical Robot Orchestra

An artist perception of Robot Orchestra



Qrio

- An invention of Sony
- Qrio— **Q**uest for cu**RIO**sity
- Smoother and faster humanoid robot than ever before
- 7.3 kg, 58 cm and 1 hour battery
- A bipedal robot which could wave hello and recognize voice

Qrio

- Could converse, sing, walk uphill, dance, kick and play using its fingers.
- Seven microphones
- Could sing in unison
- Interact with humans with movements
- Speech with more than 1000 words
- Learn new words also
- Showed emotions by flashing lights

Qrio

- Three CCD cameras in all,
- One in each eye and one at center.
- Two CCD cameras in eyes recognized up to 10 different faces and objects, and determined location of objects in view
- Four Qrios performed a complicated dance routine in 2003

Qrios of fourth generation

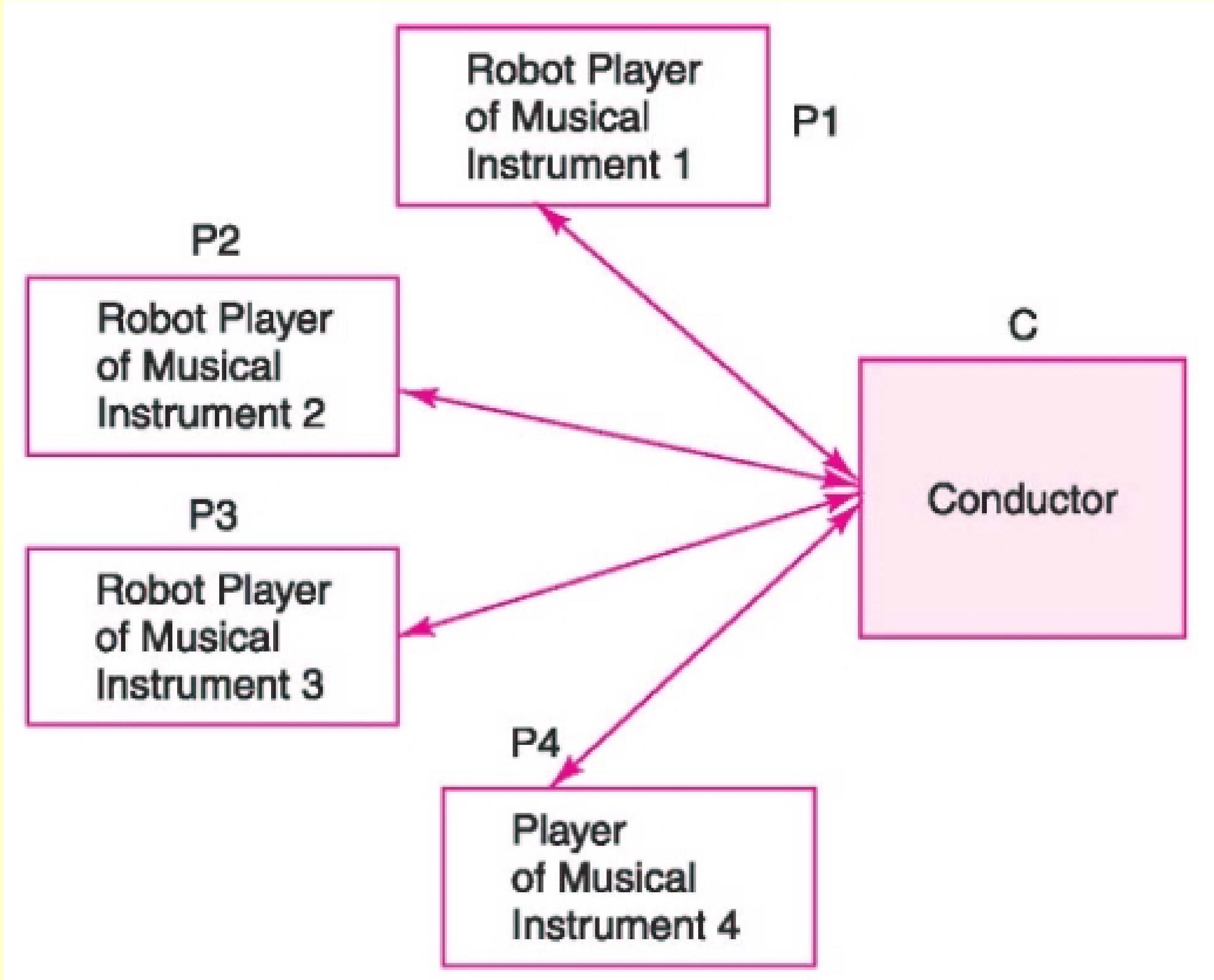
- In 2006, ten Qrios gave a dance number.
- Conducted entire orchestra.
- Played a unique rendition of Beethoven's 5th symphony (1808) http://en.wikipedia.org/wiki/Symphony_No._5_(Beethoven)
- Orchestra had novel instruments played by robotic actuators.

Qrio

- Each Qrio had 38 fluid motion flexible joints controlled by separate motors for each with ASIP in each.
- Three central system microcontrollers controlled motion, recognized speech and visual images, respectively.
- Memory 64 MB with each microcontroller

2. Communication Model for Robot Orchestra

Communication Model of Robot Orchestra



MIDI (Musical Instrument Digital Interface)

- A musical device communicates data to another using MIDI protocol
- Most musical instruments— MIDI compatible
- MIDI IN and MIDI OUT connections
- Optically isolated with the musical instrument hardware

MIDI specifications

Define

- (i) what a physical connector is,
- (ii) what message format is used by connecting devices and controlling them in "real time" and
- (iii) standard for MIDI files.

MIDI messages

- Define an event what musical note is pressed and with what speed it was pressed.
- This event is input to another MIDI receiver.
- MIDI receiver plays that note back with the specified speed.

MIDI messages…

- An entire ensemble of a robot orchestra can be controlled using MIDI protocol.
- Also the actuators are synchronized as per the notes and speeds.

A MIDI message

- Consists of a command and corresponding data for that command.
- Data sent in byte formats and are always between 0 and 127
- Corresponding command bytes in a channel message and are always are from 128 to 255

Channel message (between 0x80 to 0xEF) in MIDI file

- Musical note, pitch-bend, control change, program change and after-touch (poly-pressure) messages.

MIDI Format specifications

- Maximum 16 channels in a system
- MIDI specifies system messages, manufacturer's system exclusive messages and real time system exclusive messages (between 0xF0 to 0xFF)

Real time system message example

- A MIDI start from conductor is 0xFA command
- Always begins playback at very beginning of the song (called MIDI Beat 0)]
- So when a slave player receives MIDI Start (0xFA), it automatically resets its song position = 0

Inter-robot communication of MIDI files

- Transport of a MIDI file─ Bluetooth, high Speed Serial, USB or FireWire.
- For robot orchestra, communication protocol for MIDI files can be Bluetooth or WLAN 802.11

3. Programming model Model for Robot Orchestra

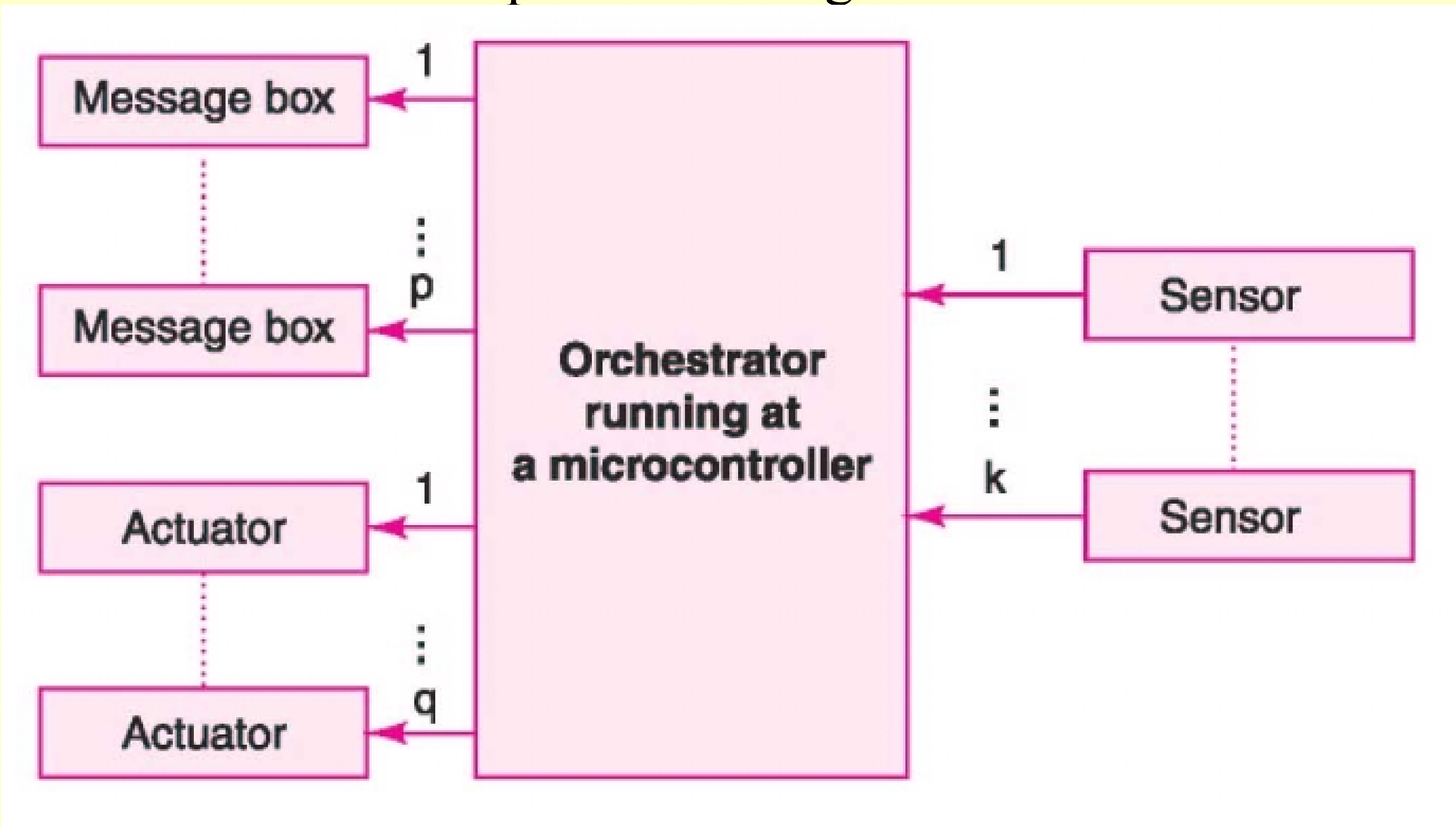
Inputs and Outputs to a software module

- *k* sensor inputs to a module
- *q* outputs generate to actuators in a sequence
- *p* outputs to message boxes (also called mailboxes in certain OSes or notifications in certain OSes) in a sequence

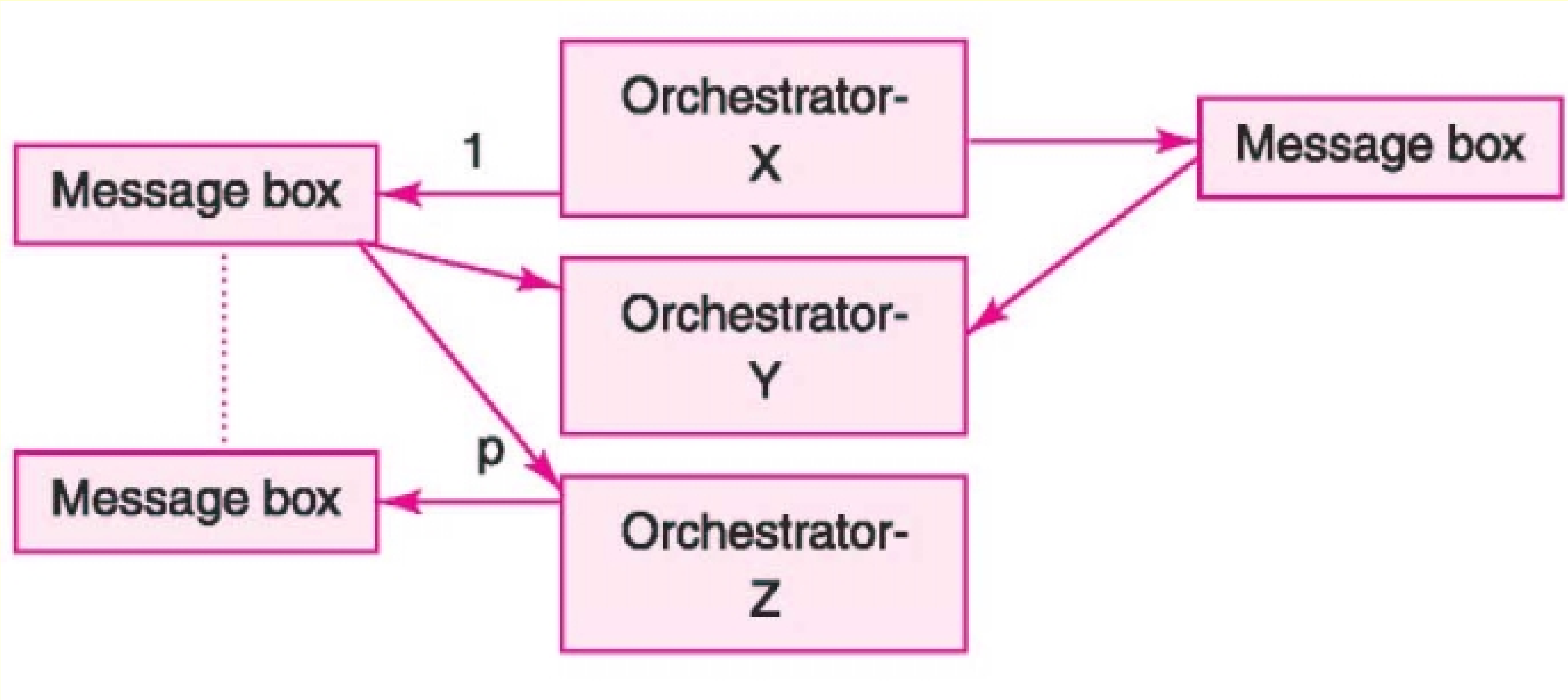
Orchestrator software module

- Software which sequences, synchronizes the inputs from 1st to kth sensors and generates the messages and outputs for the actuators, display and message boxes at the specified instances and time intervals.
- Message boxes store the notifications, which initiate the tasks as per the notifications

Program Module Orchestrator-1 at a microcontroller 1 with *k* sensor inputs and *q* outputs to actuators and *p* outputs to message boxes



Commands and messages communication between Orchestrator-x, Orchestrator-y and Orchestrator-z software modules at same or different microcontrollers



4. Requirements of Inter-robot Communication of MIDI Files

Purpose

- To communicate selected MIDI file data over Bluetooth personal area network from Robot Conductor communication task to music playing robot tasks

Inputs

- 1. Orchestra_Choice
- 2. MIDI File

Signals, Events and Notifications

- 1. Commands in the file

Outputs

- MIDI File for inter-robot communication
- Messages to actuators for movements

Functions of the system

- A user signals an Orchestra_Choice, say Beethoven's 5th symphony to start.
- The conductor C task_MIDI selects the chosen MIDI file and posts the bytes from the files in message queue for task_Piconet_Master.
- Piconet is a network of Bluetooth devices. Bluetooth devices can form a network known as *piconet* with the devices within a distance of about 10 m.

Functions of the system…

- Bluetooth piconet of network of master C, and slaves P1, P2, P3 and P4.
- task_PICONET_SlaveP1, task_PICONET_SlaveP2, task_PICONET_SlaveP3 and task_PICONET_SlaveP4 accept the messages from task_Piconet_Master.

Functions of the system…

- task_Piconet_Master posts the MIDI messages to a task_ MIDI_Slave.
- task_ MIDI_Slave posts the messages to a task_Orchestrator, which controls the motor movements of the slave robot actuators and musical note actuators.

Design metrics

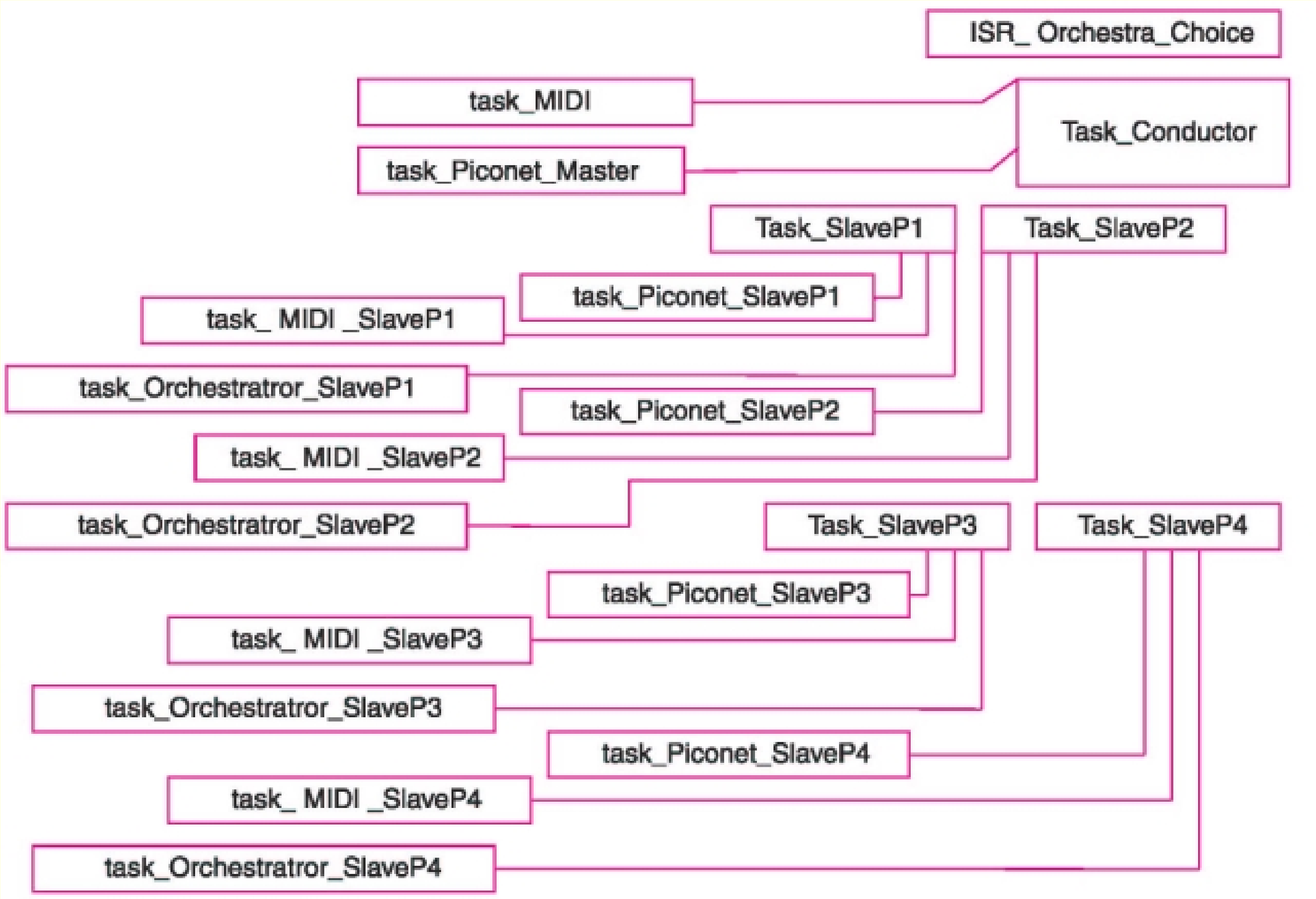
- *Power Dissipation*: As required by mechanical units, music units and actuators
- *Performance*: Human equivalent of Orchestra in music
- *Engineering Cost*: US$ 150000 (assumed) including mechanical actuators
- *Manufacturing Cost*: US$ 50000 (assumed)

Test and validation conditions

- All MIDI commands must enable all orchestral functions correctly

5. Classes and class diagram for posting MIDI file data over the Bluetooth piconet to receiving slaves

Task_Conductor class diagram



Class Task_MIDI

Task_MIDI
type0: fileType; MsgMusical Note, MsgPitchBend, MsgControlChange, Msg AfterTouch, MsgSystem, MsgManufExcl, MsgProgramChange: MsgRealTime: String NumMsg: unsigned int;
OSMsgQAccept ( ); OSMsgQPend ( ); OSMsgQPost ( );

6. Objects

Objects

- task_MIDI,
- task_Piconet_Master,
- task_Piconet_SlaveP1,
- task_ Piconet _SlaveP2,
- task_ Piconet _SlaveP3 ,
- task_Piconet_SlaveP4,
- task_MIDI_SlaveP1,
- task_ MIDI_SlaveP2,
- task_MIDI_SlaveP3
- task_MIDI_SlaveP4,

Objects (processes) of the classes Task_MIDI

- task_Orchestrator_SlaveP1,
- task_Orchestrator_SlaveP2,
- task_ Orchestrator_SlaveP3
- task_ Orchestrator SlaveP4

Message queue objects for posting and accepting the messages

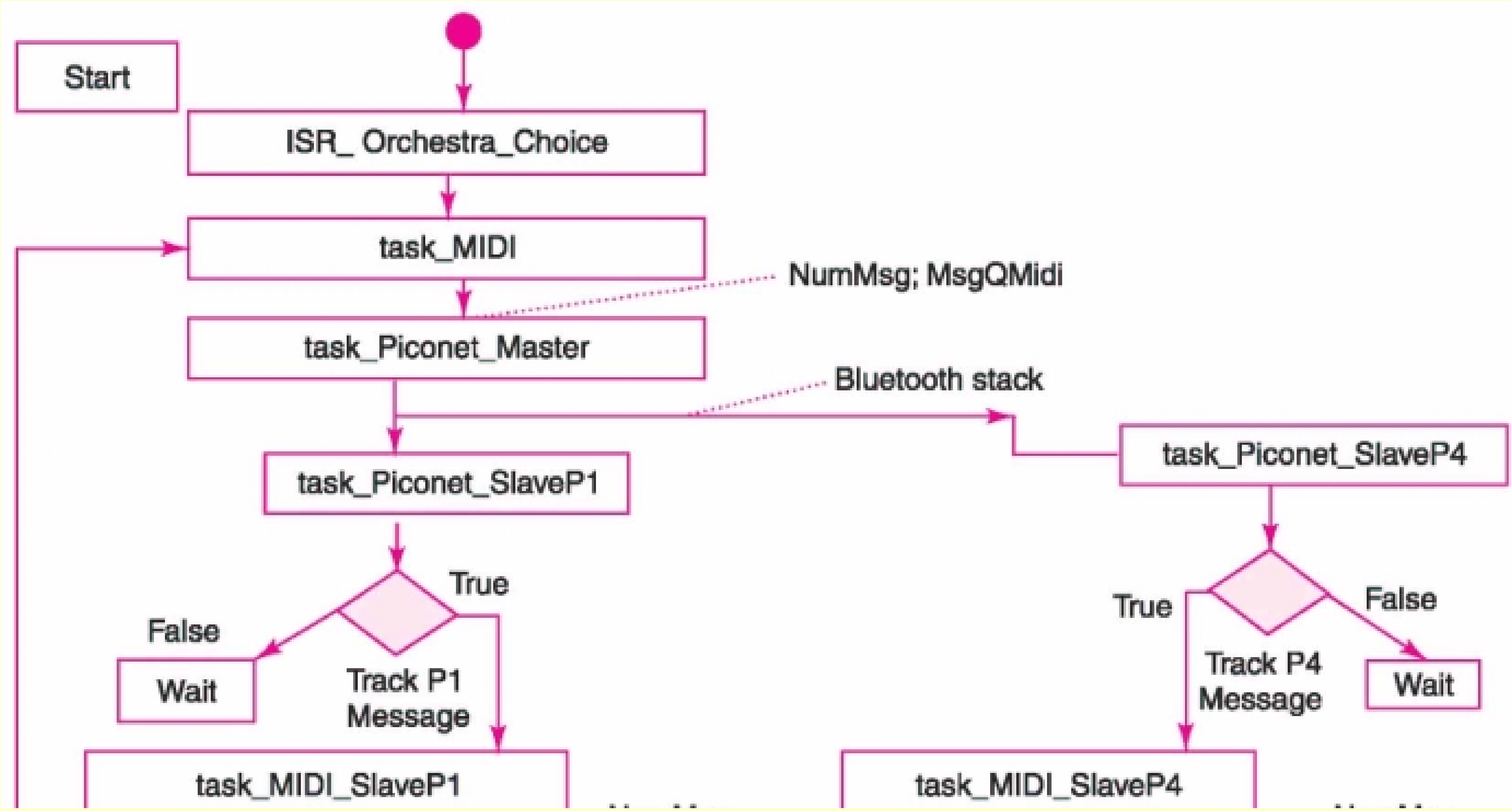
- MsgQMidi,
- MsgQBluetooth,
- MsgQMidiP1,
- MsgQBluetoothP2,
- SigPort4.

MsgQMidi

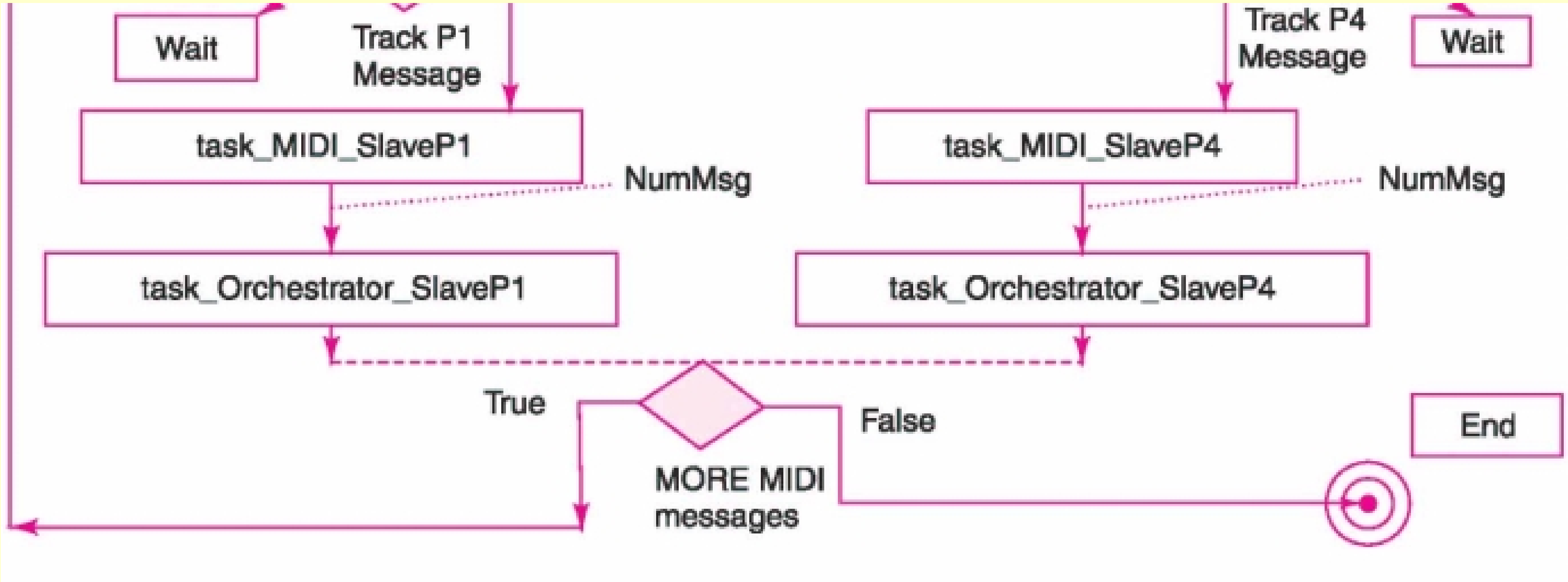
- post NumMsg messages from MIDI file in one cycle to the object task_Piconet_Master.
- MsgQBluetooth post Bluetooth stack data in one cycle to Port_Bluetooth.
- Signal object SigPort to ports initiates transfer of Bluetooth stack.
- Signal object SigPortP1, SigPortP2, SigPortP3 and SigPortP4 initiates reception of Bluetooth stack.

7. State Diagram

State diagram Part-1



State diagram Part-2



8. Hardware Architecture

Microcontroller

- A microcontroller at master and each slave to control for Orchestrator for movements.
- An ASIP for each motor movement.
- An ASIP at master and each slave for Bluetooth piconet communication between master and slaves.

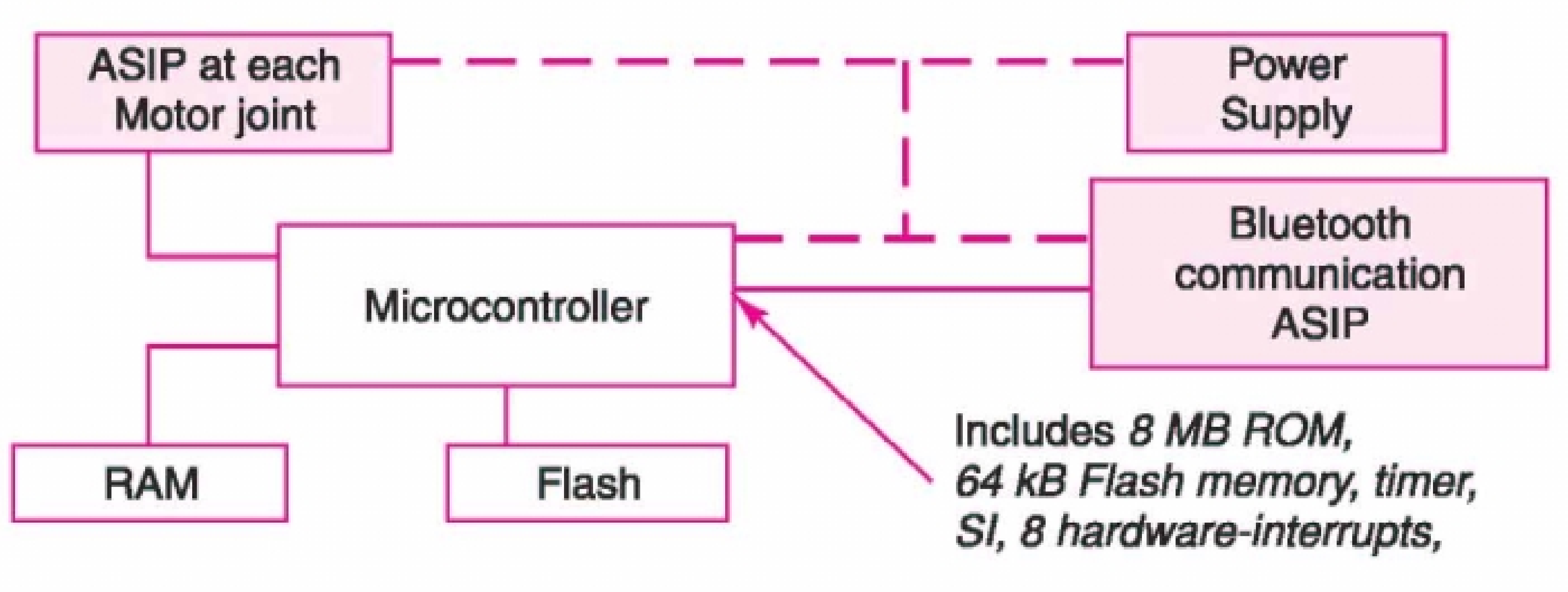
Hardware architecture

- Processors
- ASIPs
- Memory
- Ports
- Devices

Hardware architecture

- Mechanical and electromechanical units
- Interfacing and mapping of these components

Microcontroller



9. Software architecture

Software architecture

- OS
- ISRs for initiating action on user inputs and GUI notification, for example for Orchestra_Choice.
- Orchestrator tasks for master and staves.

Queue methods (functions) for system call to the OS

- OSMsgQAccept, OSMsgQPost, OSMsgPend
- Synchronize the tasks and there concurrent processing such that first task_MIDI waits for message for Orchestra_Choice.
- ISR_Orchestra_Choice codes signals and messages to Task_MIDI,

Queue methods (functions) for system call to the OS

- task_MIDI waits for NumMsg MIDI messages from MIDI file for chosen orchestra and posts to the NumMsg number MsgQMidi messages to task_Piconet_Master.
- task_MIDI posts the bytes in message queue for task_Piconet_Master

task_Piconet_Master

- Discovers the slaves and sets up piconet Bluetooth device network on first initiation.
- On accepting MsgQMidi messages, it sends protocol stack outputs through the port.
- A task_Piconet_SlaveP1 sets up network with master as the server.

task_Piconet_Slave

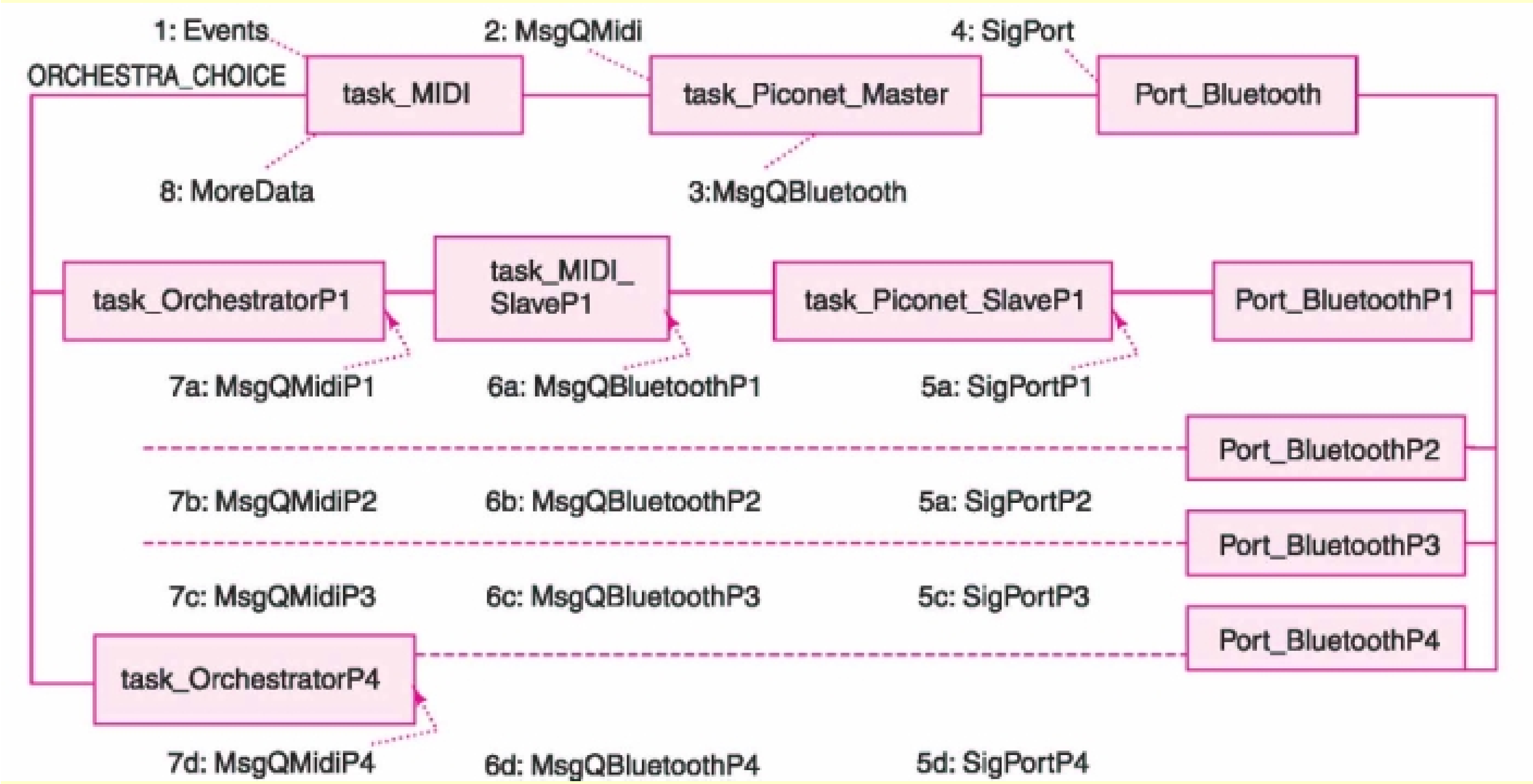
- Receives Bluetooth stack for the NumMsg messages of MIDI file and sets up network with master as server.
- The MIDI messages are sorted for the messages to be directed to track P1 or P2 or P3 or P4.

task_Piconet_Slave…

- trackP1 messages post to task_MIDI_SlaveP1 from task_Piconet_SlaveP1.
- Then task_MIDI_SlaveP1 posts the messages to actuators and Orchestrator task_OrchetratorP1.
- Step similar repeat for other slaves P2, P3 and P4 tracks in robot orchestra.

10. Multiple tasks and their synchronization model

Synchronization model for master-slave robots communication tasks



Summary

We learnt

- Case study for the inter-robot MIDI file communication
- Orchestrator concept
- Requirements ,
- Class diagrams, classes and objects
- State diagram
- Hardware and software architecture
- Tasks synchronization model

End of Lesson-1 of chapter 12 on Case Study of Inter-Robot Communication in a Robot Orchestra