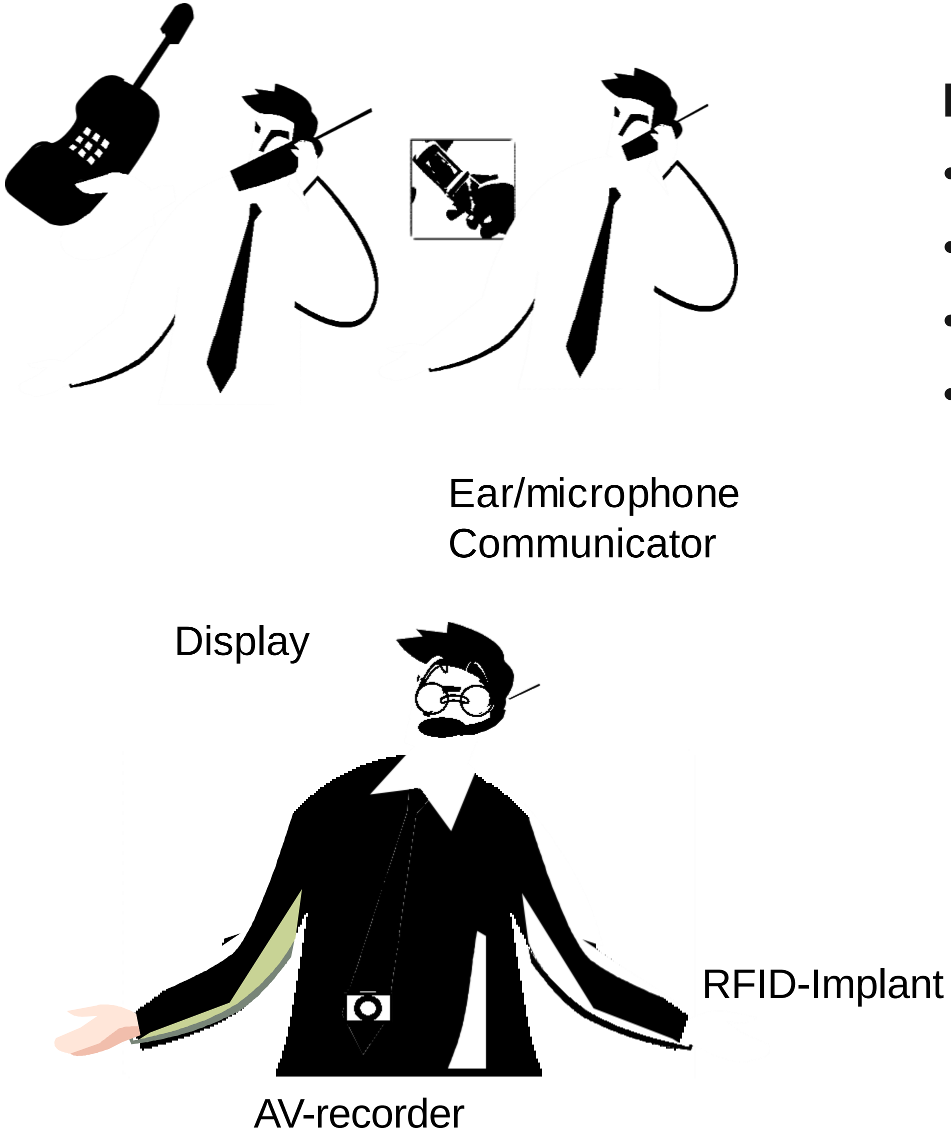
Pervasive Computing
Lecture – Augmentation, Design Principles, Privacy and Safety

Platforms

- Augmentation of Society and Life
- Design of Pervasive Systems (HCI)
- Safety and Security

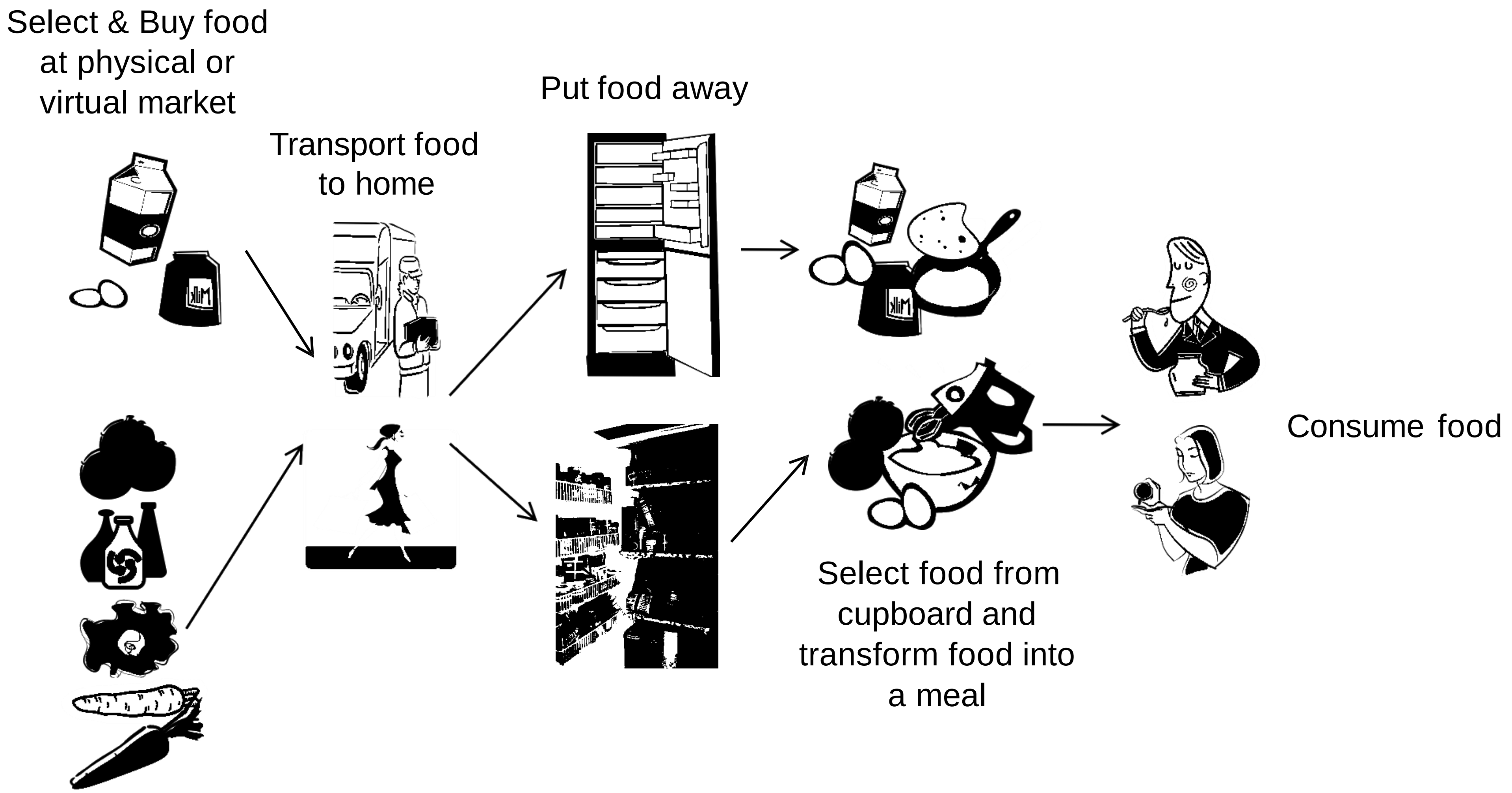
Augmentation



Reasons:

- Need – Support and Healthcare
- Control – Surveillance
- Exploration
- Transhumanism

Augmentation



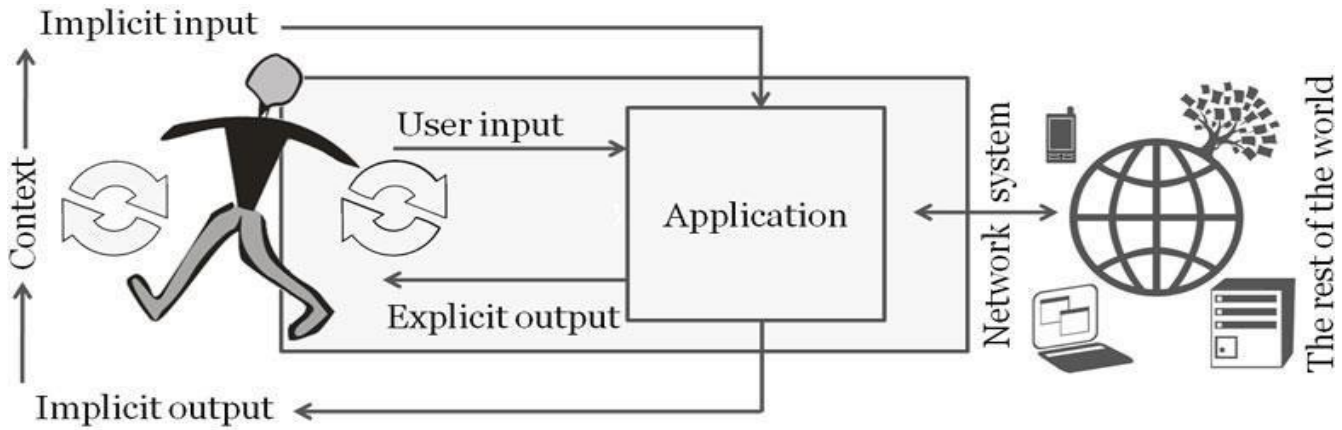
Design

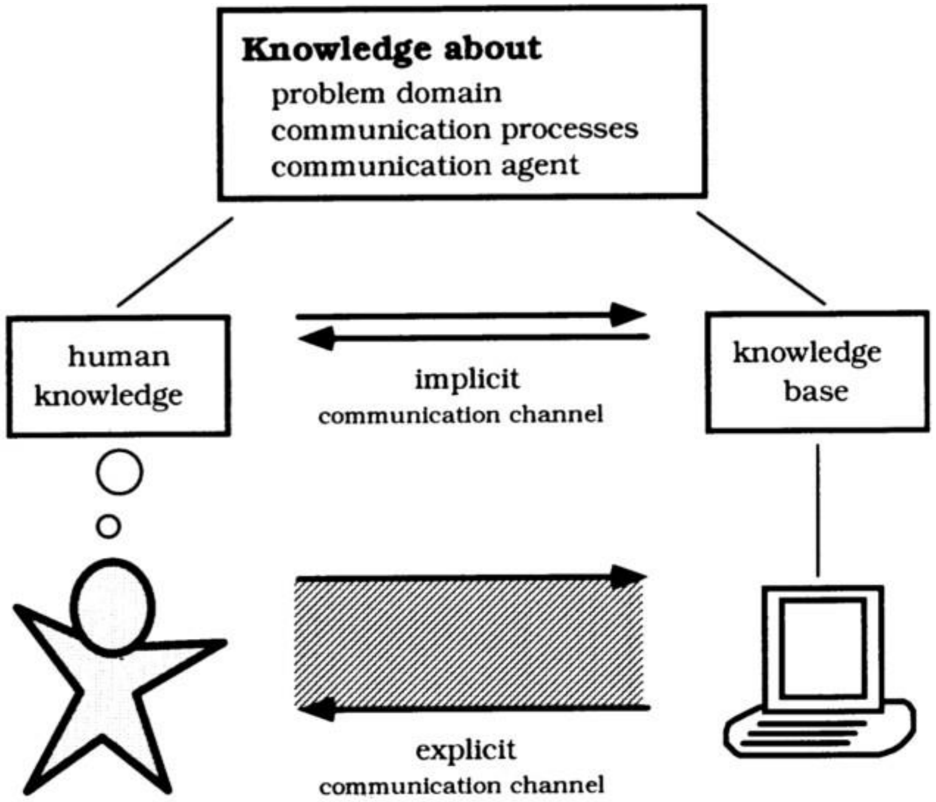
- Regular interface design is not sufficient
- HCI (Human-Computer-Interaction)
  - Explicit HCI
  - Implicit HCI

- Regular interface design is not sufficient
- HCI (Human-Computer-Interaction)

  - **Explicit HCI:**
    - System performs actions based on user input
    - Feedback is anticipated by the user
    - Designed for specific context
  - **Implicit HCI:**
    - System performs actions based on human actions (including input)
    - Feedback based on anticipated human needs (beyond user input)
    - Design for multiple contexts

Design

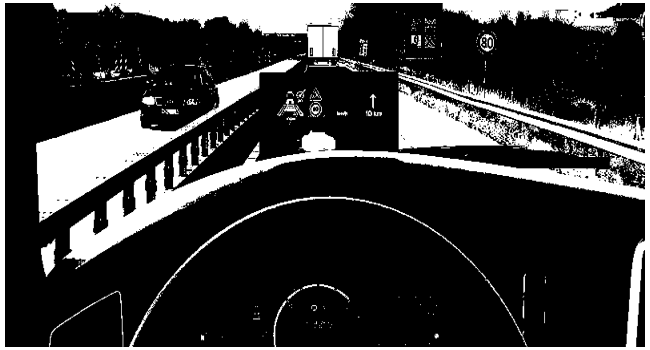


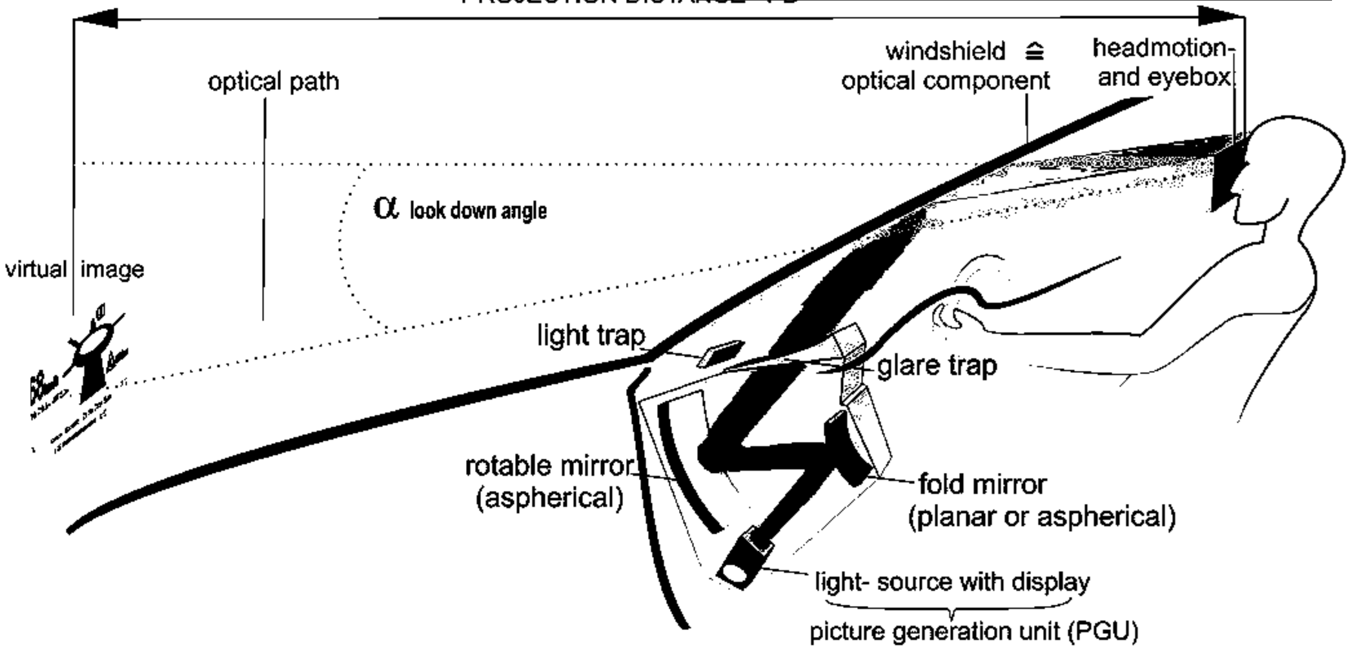


.gure 5. Knowledge based Human computer Interaction [21].

Safety and Security

- Safety:
  - Difficult as systems are not in controlled environments
  - Context important / changes
  - Which information to trust
  - How to deal with errors





livio AI

Safety and Security

- Safety and Privacy:
  - Large number of Regulations:
    - UK Data Protection Act (2018)
    - US Data Privacy Act (1974)
    - GDPR – General Data Protection Regulations (2018)
      - Protect privacy of EU citizens
      - Affects all interactions with EU citizens and their data
      - Core principles: lawfulness, fairness and transparency
      - Requires compliance or will face fines

Safety and Security

- Anonymity:

  Effective Technical Solutions for Anonymous Communication

    Mixes, Proxies, e-Cash,…

  However, many services require or perform some form of identification

    Customisation, Delivery, Cameras,…

  Pseudonymity can be a good substitute

    Can be thrown away, though often used Pseudonyms may become valuable

    Do Pseudonyms have a right to privacy?

  Data Mining may find ”real” identity!

Safety and Security

- Security:

Secure Communications

Gets the information safely across

Secure Storage

Locks the information safely away

Usage?

Transparency

What do they do with the data?

Recipients?

Who gets the data?

Retention?

How long do they keep the data?

Safety and Security

- Trade-offs:

Convenience vs. Anonymity

The more the others know about me, the better they can accommodate my preferences

Personal Liberty vs, Social Utilitarianism

Increased Surveillance for apprehending criminals

Success Rate vs. Risk of Failure

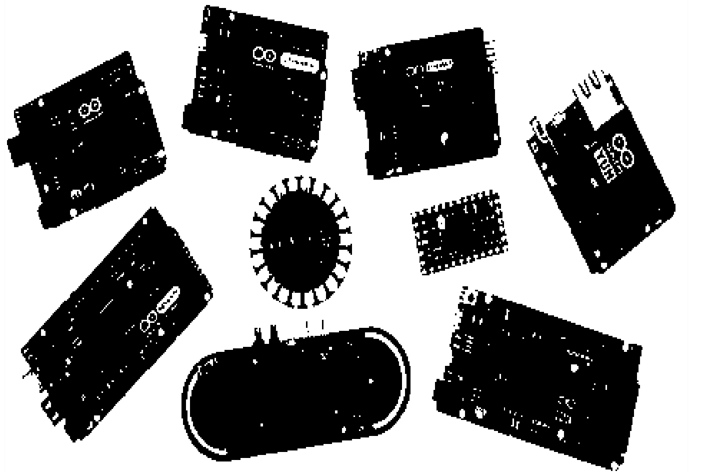
Pervasive Games

- Bridge between physical-virtual
- Difficult to design; hard to describe
- Future of entertainment?
- Rely on:
  - Physical (inter-)action
  - Current technology
  - Broad range of users/interactions
  - Heavy reliance on infrastructure and persistent internet

- Download either:
  - Android: Sensor Data Logger
  - iOS: Sensors – Sensory i czujniki -

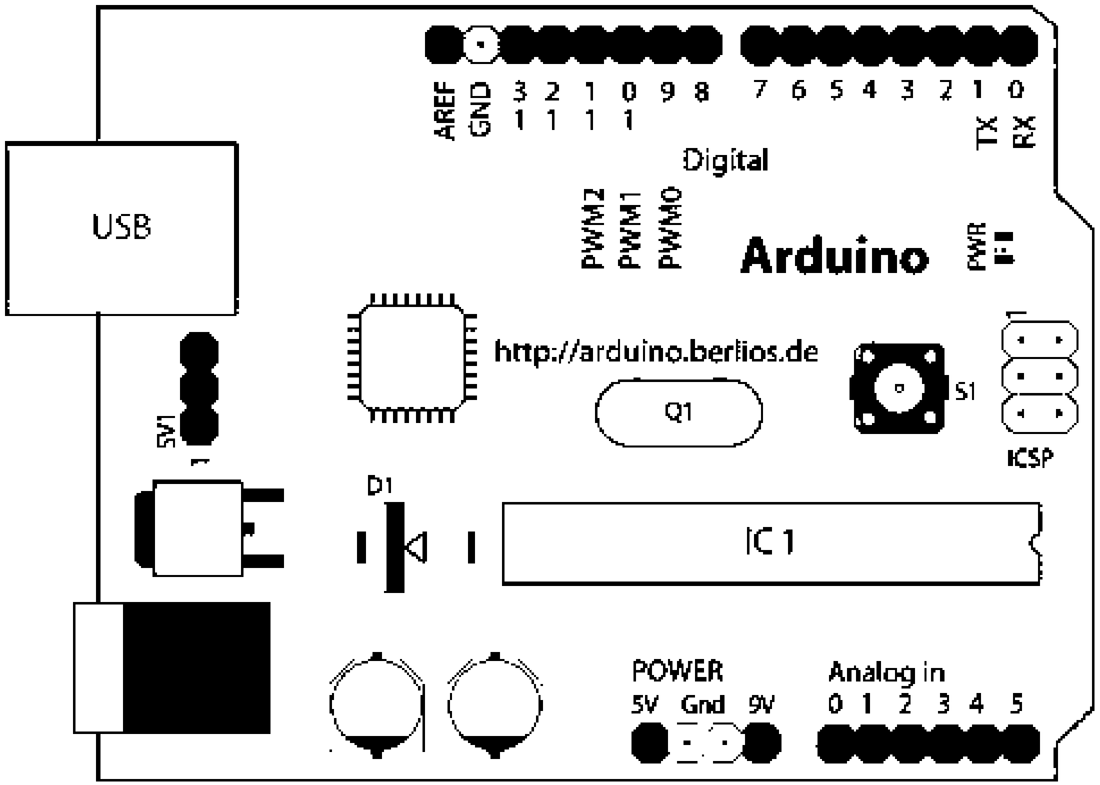
Pervasive Computing
Lecture 5 – Platforms, Data Structures & Sockets

Platforms



Arduino

- Microcontroller
- 32kB/16MHz (Uno)
- Versatile
- Extensible
- Low power consumption
- Low computing power
- Wide variety of form factors
- Analog/Digital pins
- Open source hardware/software

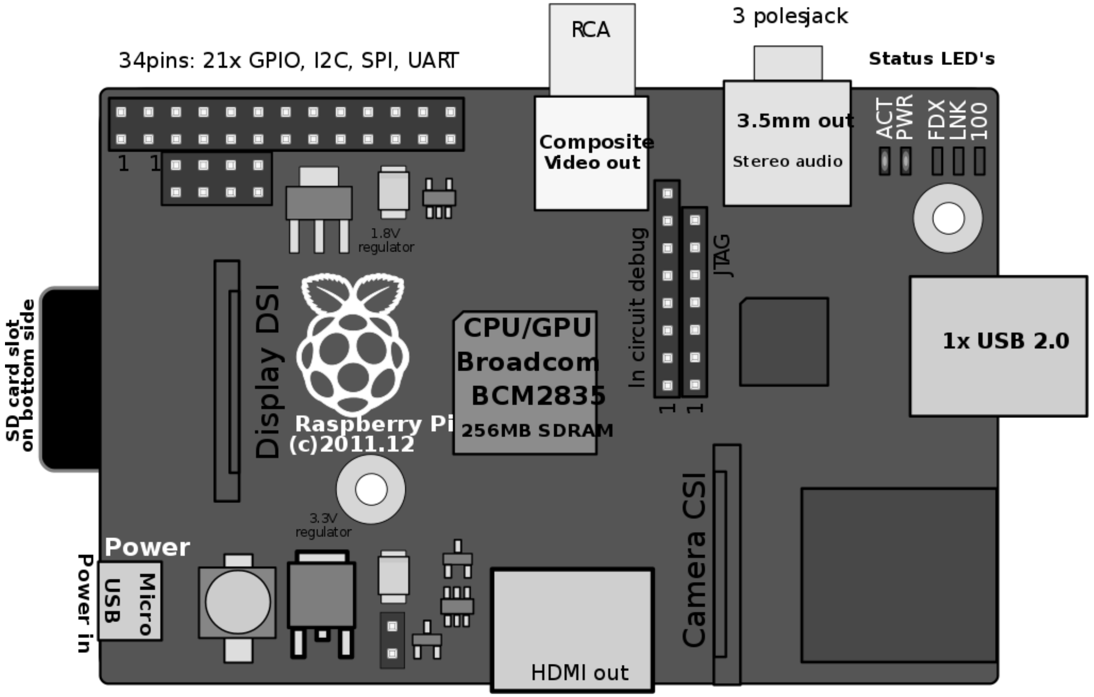


Platforms

Raspberry Pi

- Linux based embedded PC
- Provides networking capability
- Moderate power consumption

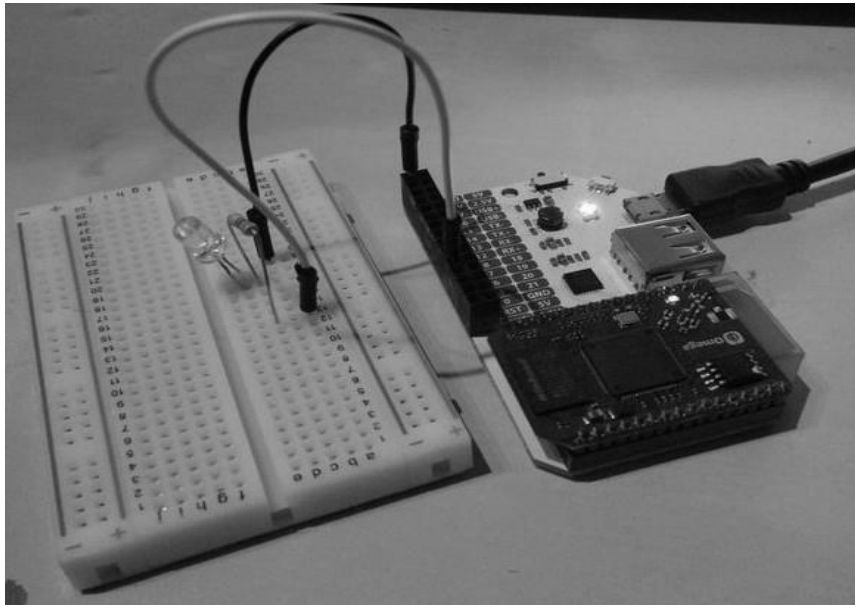
  - Moderate computing power
  - Expandable memory
  - Digital pins
  - High-level language support
  - Licensed hardware



Platforms

Onion Omega

- Linux based embedded PC
- Good memory
- Variety of bases
- Tiny footprint
- Internet/network focused
- Low power consumption
- Inexpensive
- New platform (risk?)
- High-level language support



Python

- Structure
- Data Structures
- Sockets
    - Multi paradigm language (idiom pythonic)
        - Code readability / syntax
        - Modern language (1991 released)
    - Easy entry and wide availability
    - Interpreted language

Python Data Structures

List:

- Versatile
- Standard

```
a = [1,2,4]
print(a[2])
>>> 4
a.append(5)
Print(a[3])
>>> 5
```

Tuple:

- Immutable
- Fast
- Can be concatenated

```
a = (1,2,4)
print(a[2])
>>> 4
a += a
print(a)
>>> (1,2,4,1,2,4)
```

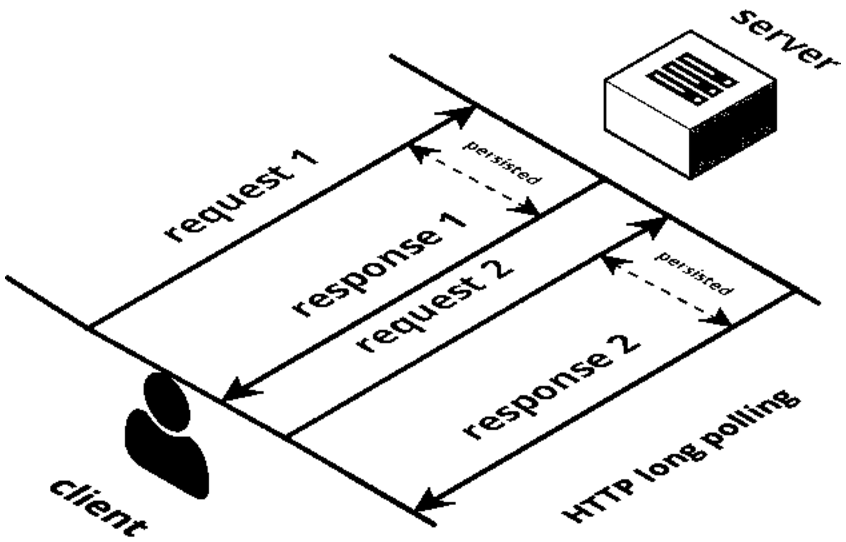
Dictionary:

- Key-value pairs
- Standard

```
a = {‘1’:1,‘2’:2,’3’:3}
print(a[‘2’])
>>> 2
a.keys()
a.values()
```

Python Socket - Background

- Originated with ARPANET in 1971
- Later became an API in 1983
- Network programming took off in the 1990s

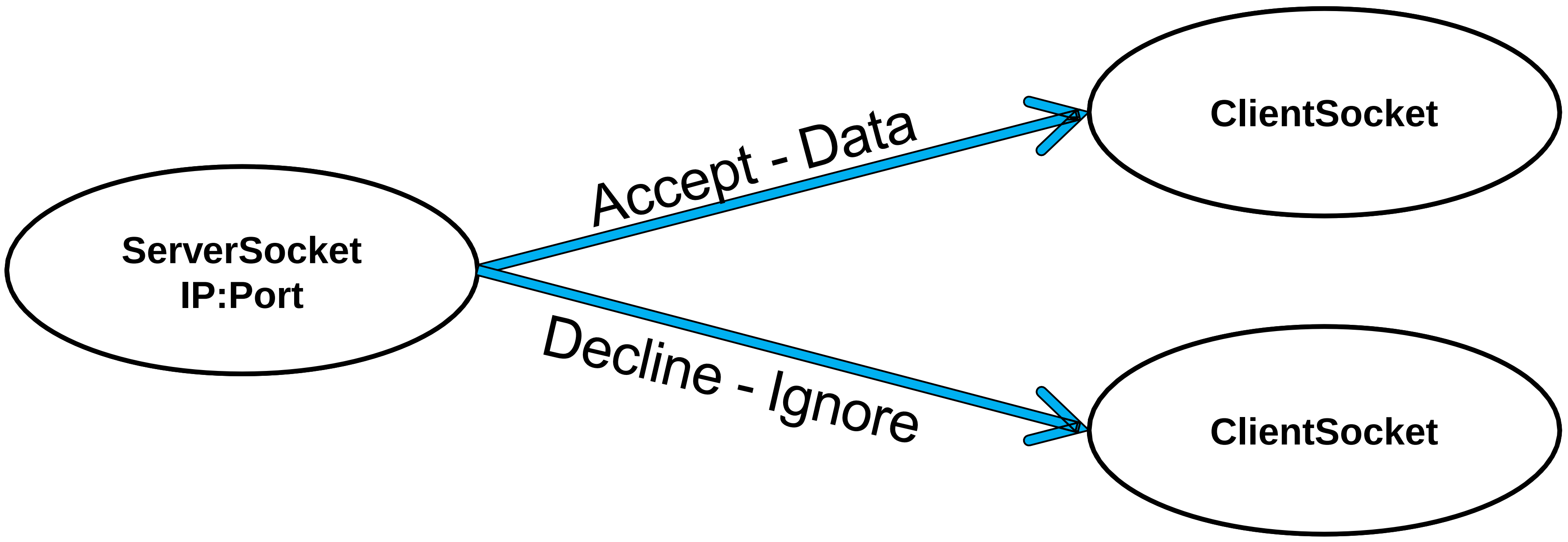


Python Socket – Exchanging Data

- Sockets can send any data
- Which data do you expect?
- Data exchange format

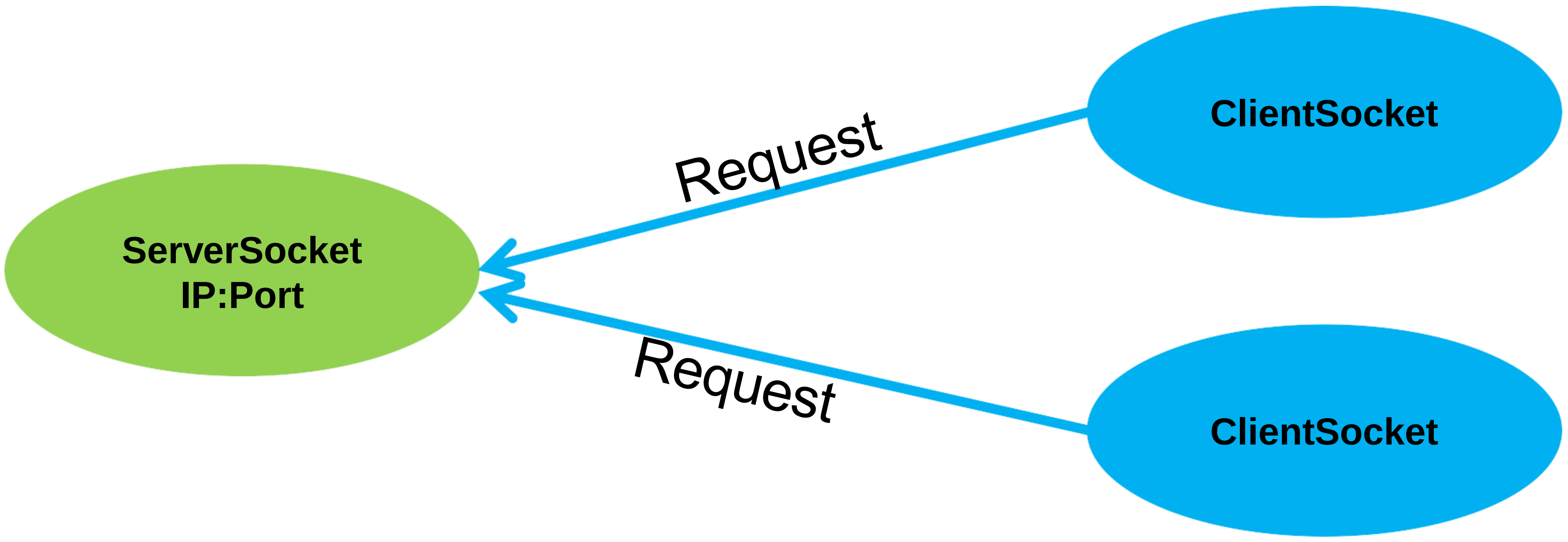
Python Socket - Server

- Low-level network interface
- Bi-directional communication (one to n)
- host:port



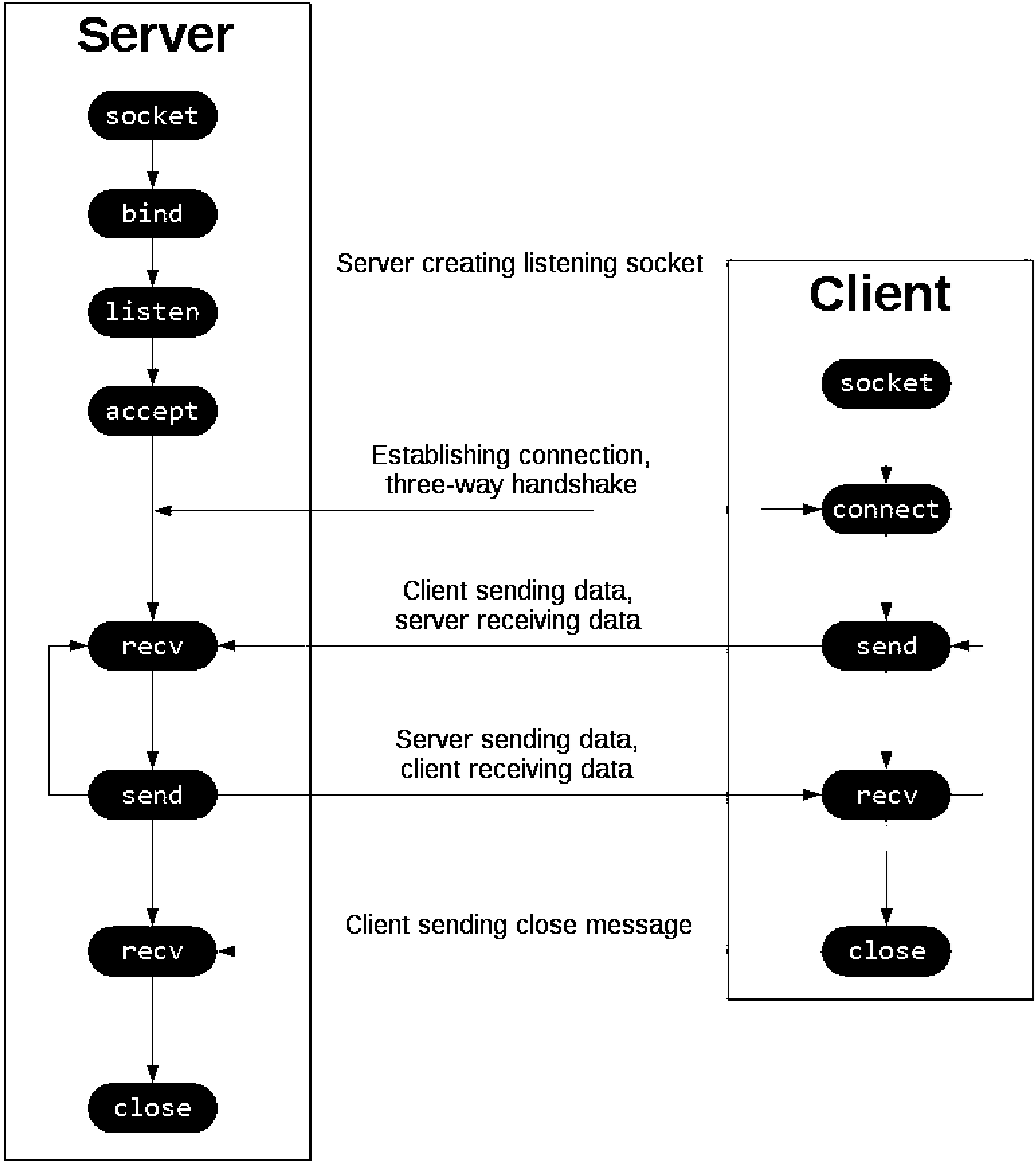
Python Socket - Client

- Low-level network interface
- Bi-directional communication (one to n)
- host:port

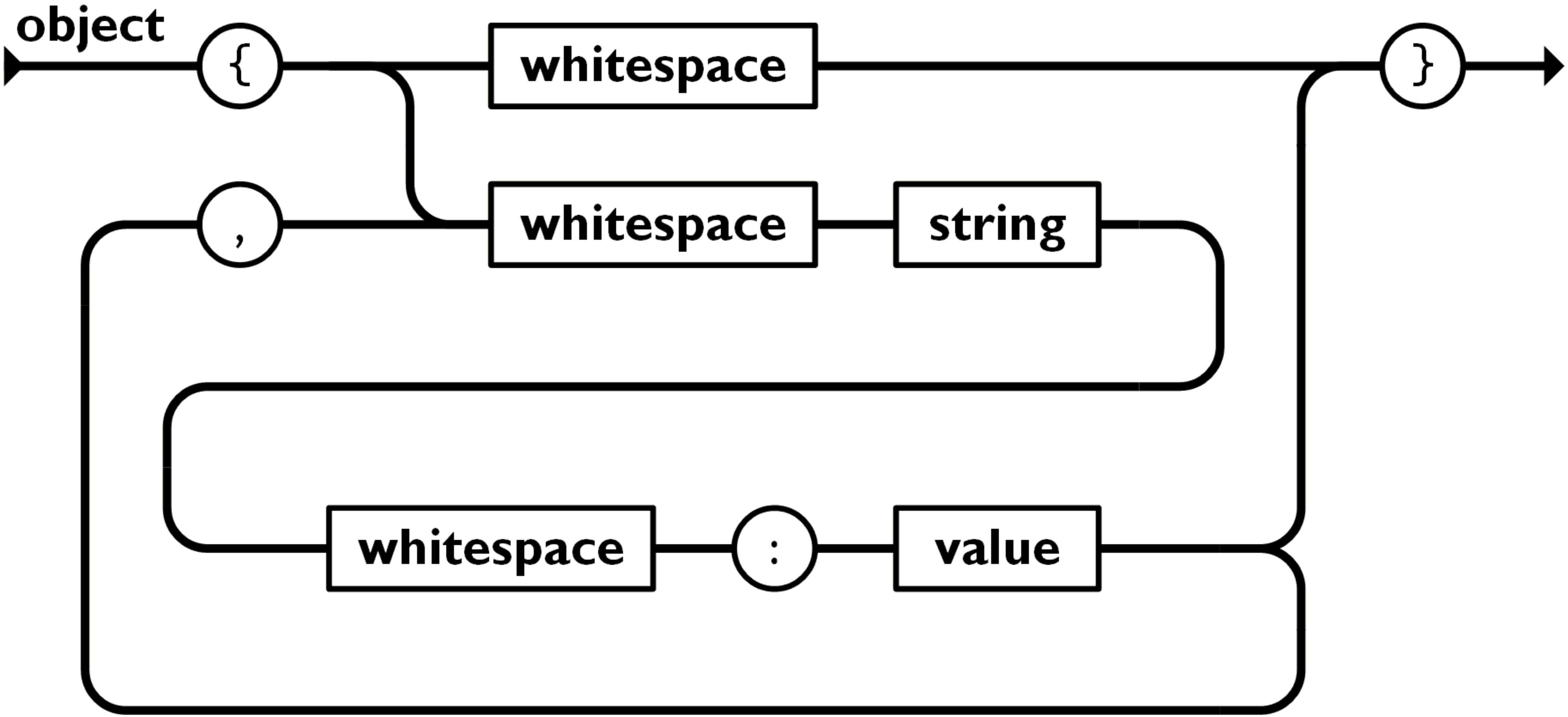


Python Socket

- Protocol: TCP, UDP
- Waiting for data →processing
- Complex:
  - Synchronisation
  - Error handling
  - Data types



Python Socket – JSON



Python Socket – JSON

- JSON (JavaScript Object Notation)
- Light-weight format
- Cross-platform JSON parsers
- Easy to use and robust

Client

```
import json
import time

data = {‘sensor’:’sonic’, ‘value’:110,’time’:time.ctime()}
raw_data = json.dumps(data, sort_keys=True, indent=4)
```

Server

```
import json
import time

raw_data
data = json.loads(raw_data)
```

- Issues:
  - Complex objects
  - Nesting of data
  - Looping references
  - No error handling