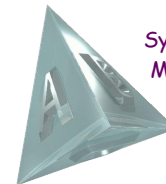


Homogeneous Resource Configuration and Access for an Autonomous Robotic Vehicle

Steven D. Johnson, Bryce Himebaugh & Scott A. Dial

Computer Science Department
School of Informatics
Indiana University
sjohnson@cs.indiana.edu



System Design
Methods Lab



Embedded & Real-Time Systems Lab

Background

- Participation in DGC entry development
 - 13-month consortium effort [Indy Robotics]
 - individuals, groups, small companies, university groups
- Lessons:
 - reasonable infrastructure (Pub/Sub)
 - system integration problems (recalibration, asynchronous)
 - distance collaboration is difficult
 - high general interest at IU---cognitive robotics, computer learning, networking, etc.
- Local *experimental platform* needed

ERTS Embedded & Real-Time Systems Lab

Functional goal:
autonomous real-world navigation

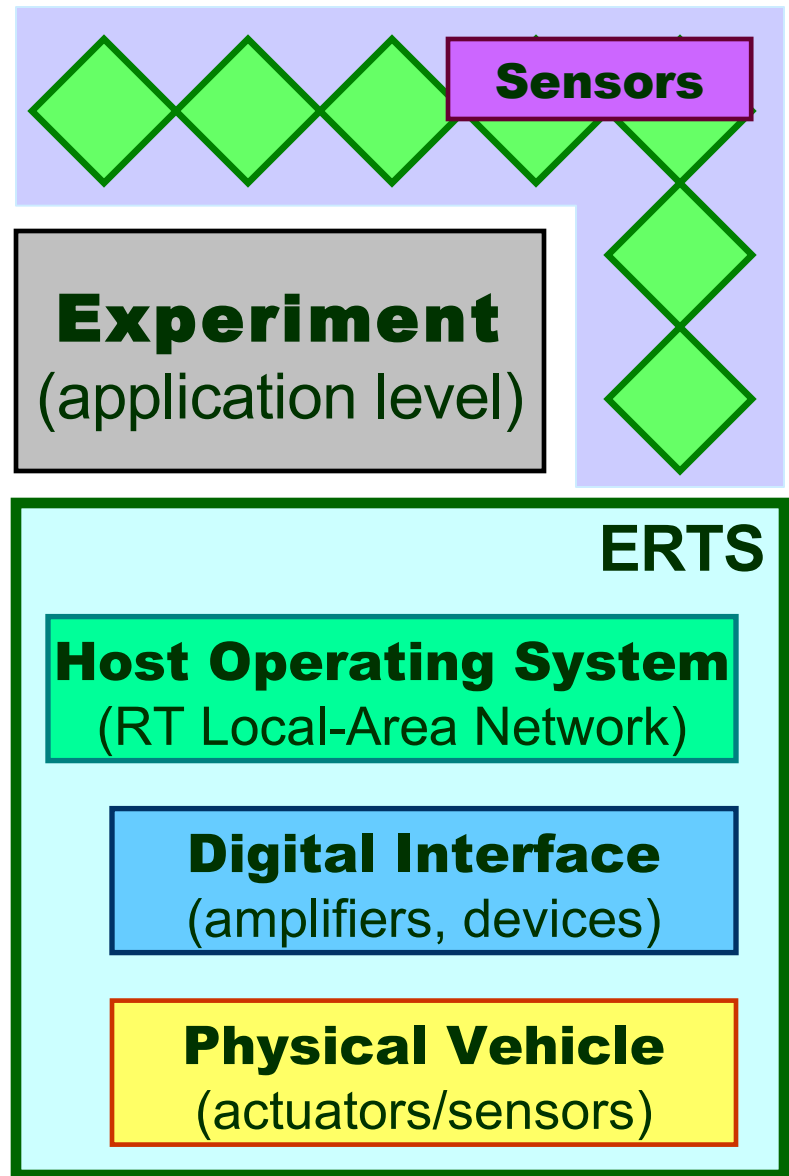
Mission: *collaborative* research and instruction

Needs: horizontal and vertical *configurability*



ERTS Architecture

- ➔ CS Systems Research
 - deep penetration
- ➔ AI, Vision, CogSci
 - flexible capabilities
 - rapid integration
- ➔ Education
 - early (ugrad.) exposure
 - physical testing
 - diverse backgrounds
 - *rapid* prototyping



SyncFS Synchronized File System

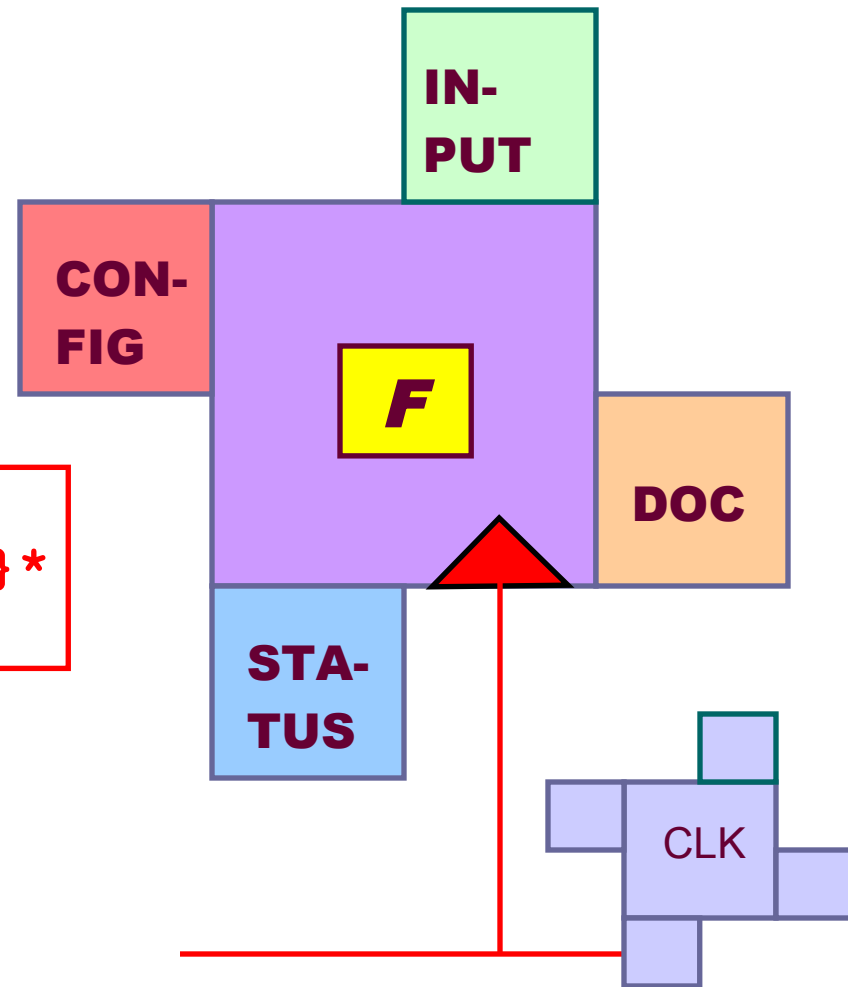
- ▶ Modified version of P9 NFS from [Plan 9] inspired by [Brown & Pisupati, 2006]
 - small, open source, ships with Linux
- ▶ Basic FS I/F to shared data (asynchronous)
 - interal files
 - **mount -t 9p ip dir -o rw,port=p,access=any**
- ▶ Synchronous support
 - **CLK** component
 - **write, stat** deferred to CLK "edge"

CartFS Component Model

- MRSW file system
- Component
 - synchronizes with CLK
 - writes only its STATUS

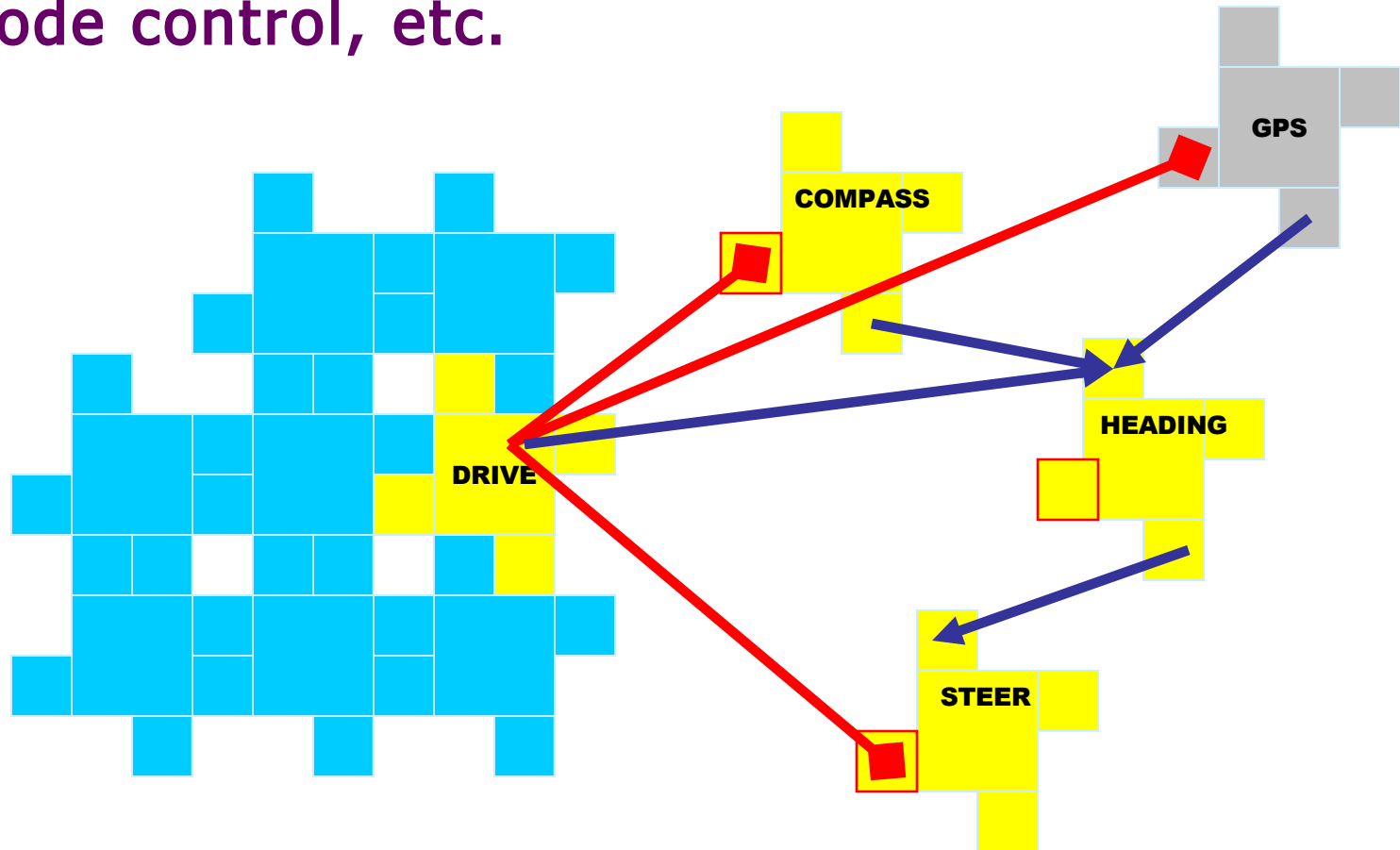
```
init;  
{read ; process ; write; }*  
exit;
```

- file \approx association list

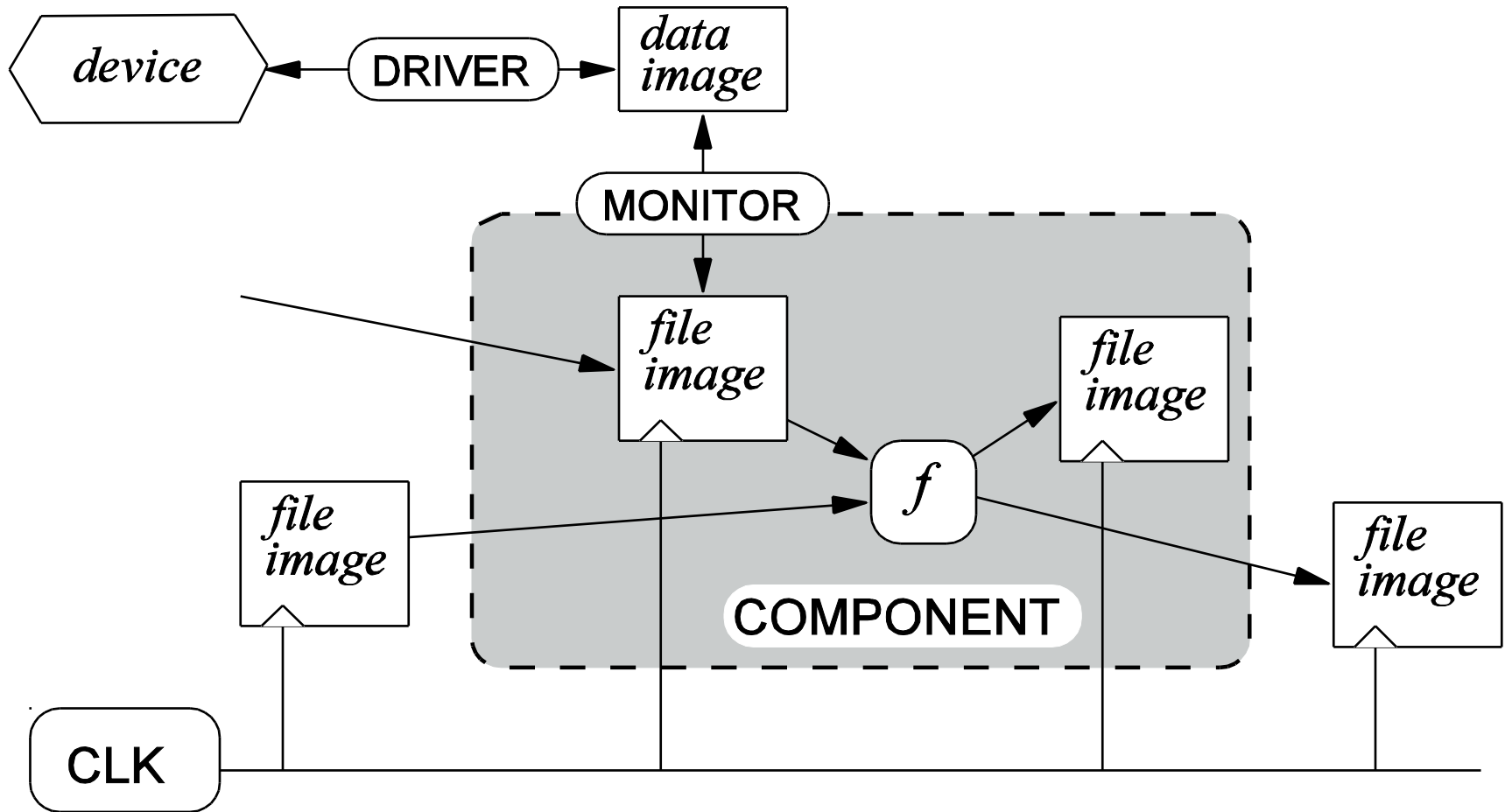


Dynamic configuration

- ◆ Call by reference (file name) via CONFIG file
- ◆ Mode control, etc.



CartFS component



Conclusions, directions

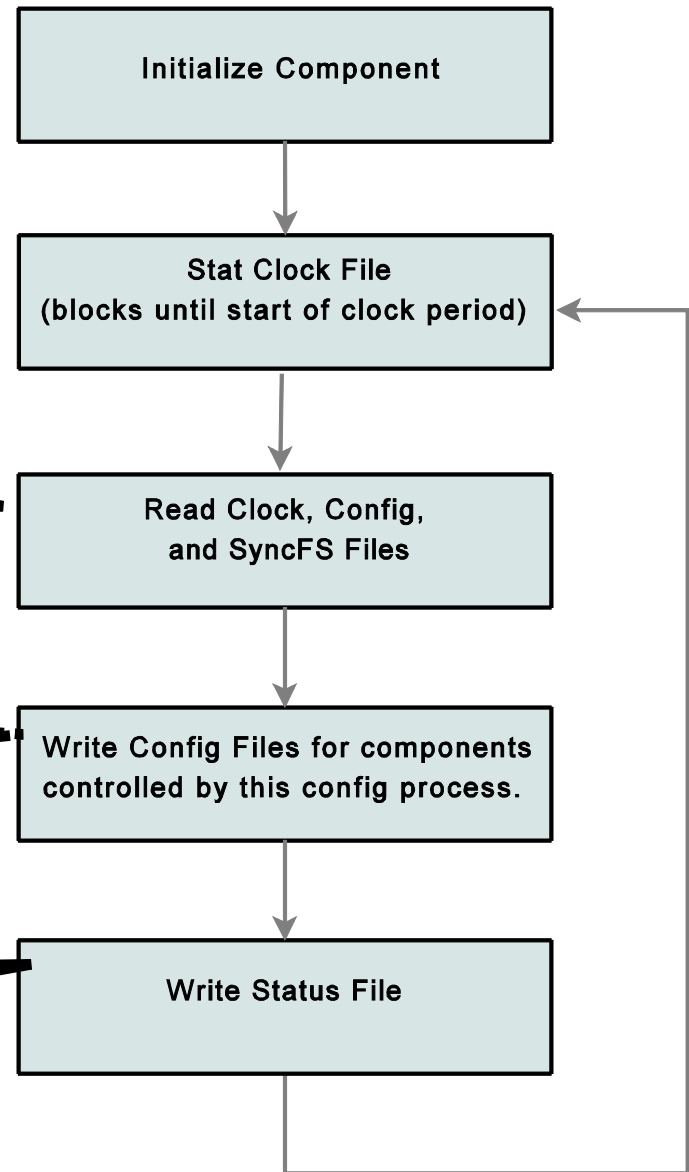
Componet flow

synchronize

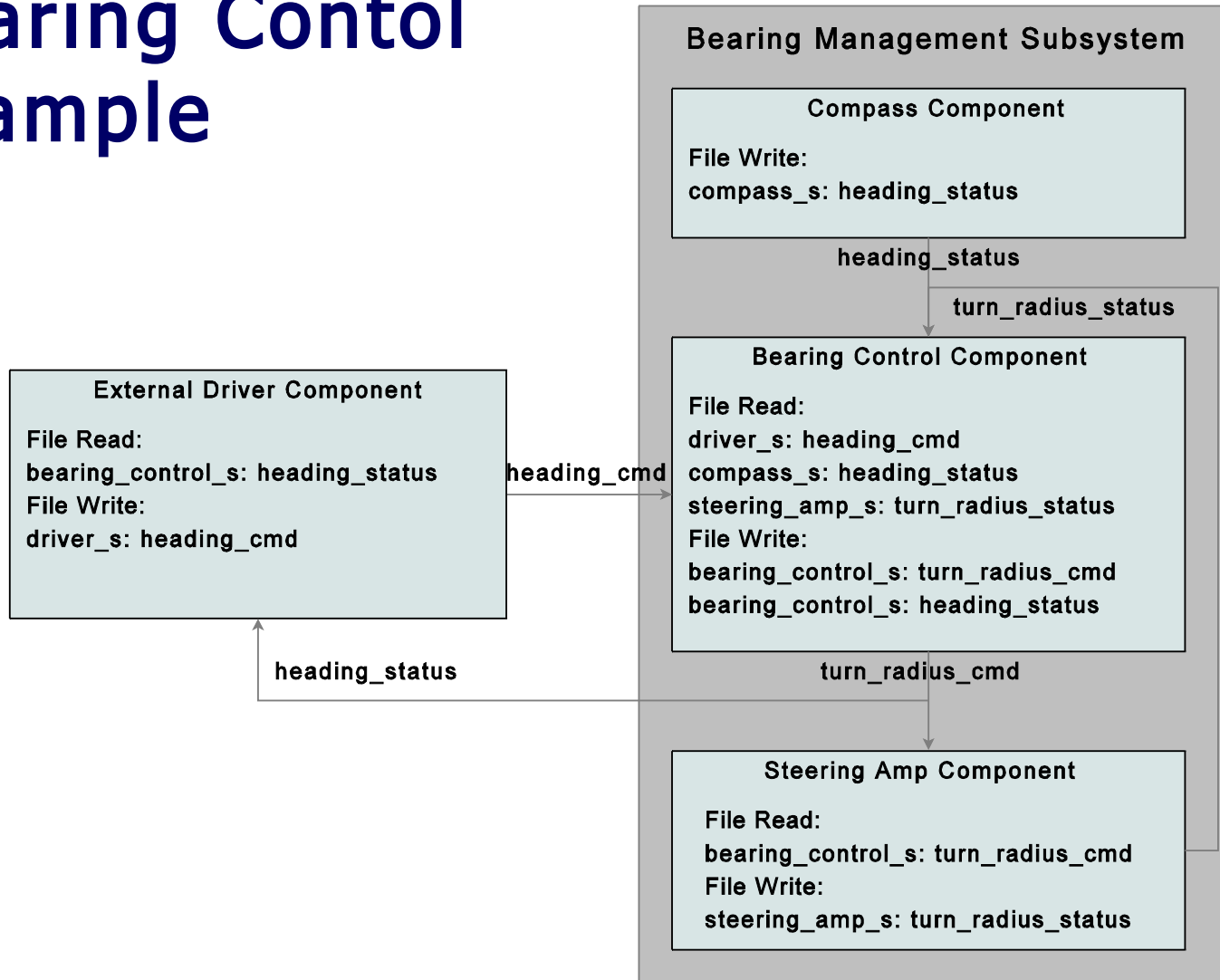
**control & data
inputs**

**configuration
& command**

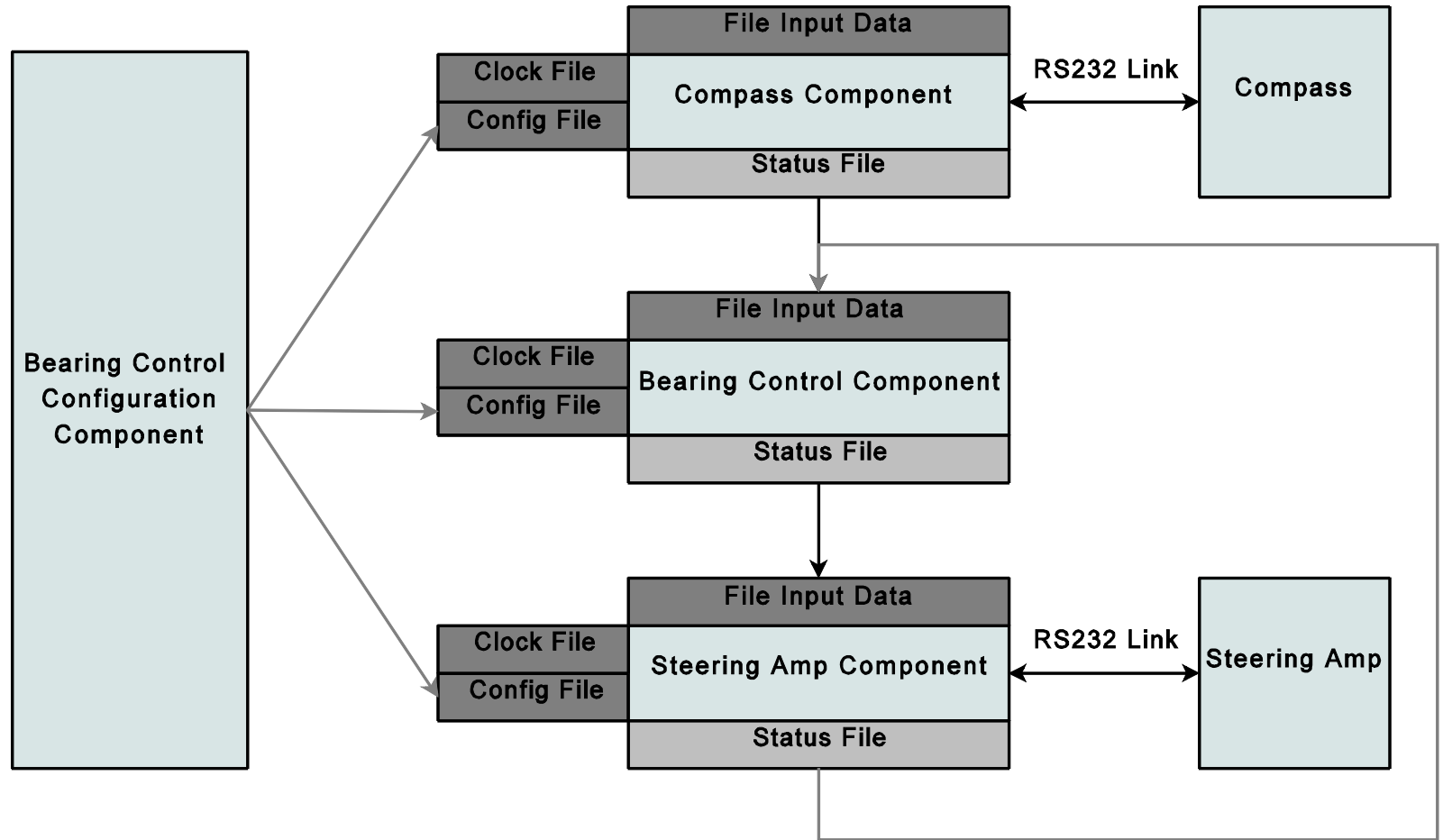
**configuration
& command**



Bearing Control Example



Bearing Control configuration



```
#!/usr/bin/env python
[REDACTED]
from cartfs import Sensor
class options:
    root = '/tmp/cartfs`
    clock = 'clock`
    port = '/dev/cartfs_compass`
[REDACTED]
class CompassHandler():
    def get_heading(self):
[REDACTED]
class CompassSensor(Sensor):
    def process(self):
        if self.compass_c['enable'] == 'True':
            self.compass_s['heading'] = self.compass.get_heading()
        else:
            self.compass_s['heading'] = "Disabled"
            self.compass_s['enable'] = self.compass_c['enable']
[REDACTED]
```

Driver.py

```
#!/usr/bin/env python

from cartfs import Sensor
class options:
    root = '/tmp/cartfs'
        clock = 'clock'
        config = './driver/driver_c'
        status = './driver/driver_s'

class DriverSensor(Sensor):

    driver_c = {'enable':True}
    driver_s = {'clock':0, 'enable':True, 'desired_heading':0}

    def process(self):
        self.driver_s['desired_heading'] = self.desired_heading
```

headingcontroller.py

```
#!/usr/bin/env python
from cartfs import CartFSLoggingHandler, Sensor

class HeadingcontrollerSensor(Sensor):
    headingcontroller_c =
        {'compass_heading':
            ['/tmp/cartfs/compass/compass_s', 'heading', -1],
        'desired_heading':
            ['/tmp/cartfs/driver/driver_s', 'desired_heading', -1]}

    def process(self):
        self.headingcontroller_s['inv_turn_radius'] =
            self.get_inv_turn_radius(
                self.headingcontroller_c['compass_heading'],

                self.headingcontroller_c['desired_heading'])
```

cartfs.py

```
from copy import copy
# import demjson
import cJSON
import logging
import os
```

```
__all__ = ['CartFSFile', 'CartFSLoggingHandler', 'Sensor']
```

```
# json = demjson.JSON(compactly=False)
```

```
class CartFSFile(object):
```

```
    def __init__(self, file, mode='r', create=False):
        """
```

```
        Initialize the `CartFSFile` object. Open a file for
        reading or writing
        and possibly create the file.
```

Parameters:

- `file`: a string, the name of the file to open.
- `mode`: a string, one of:
 - 'r': open the file for reading.
 - 'w': open the file for writing.



08CV-0243