Progress Report Notebook

2021-2022

Robin Rai

University of East Anglia

08-Sep-21

22-Sep-21 1st meeting date

Plan Of The Last Week

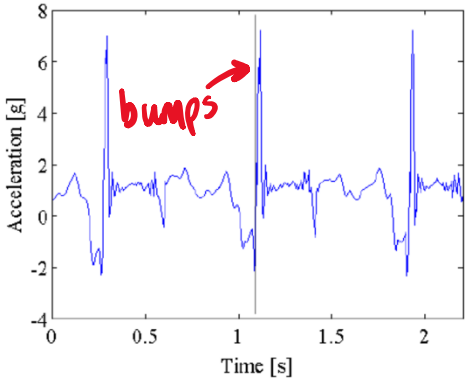
- Use this template for research notes
- What problem does this project attempt to solve?
- Why is the research problem important?
- A rough idea about how the problem to be solved?
  - Try to survey some related work
  - Go to *Google scholar* and search key words
  - Make some slides to introduce these existing solutions and point out their cons and pros
- One meeting per two weeks (say 2:30-4pm Wed)

Mountain bikes

- Feature suspension for better negotiating rough terrain
- This saps pedalling power on flat ground!
  - Some energy is wasted compressing the suspension instead of pushing you forward!
- Suspension usually comes with a lever to disable the suspension (lockout)
  - A bit of a pain having to reach down in two places all the time – especially over rough terrain

Embedded solution

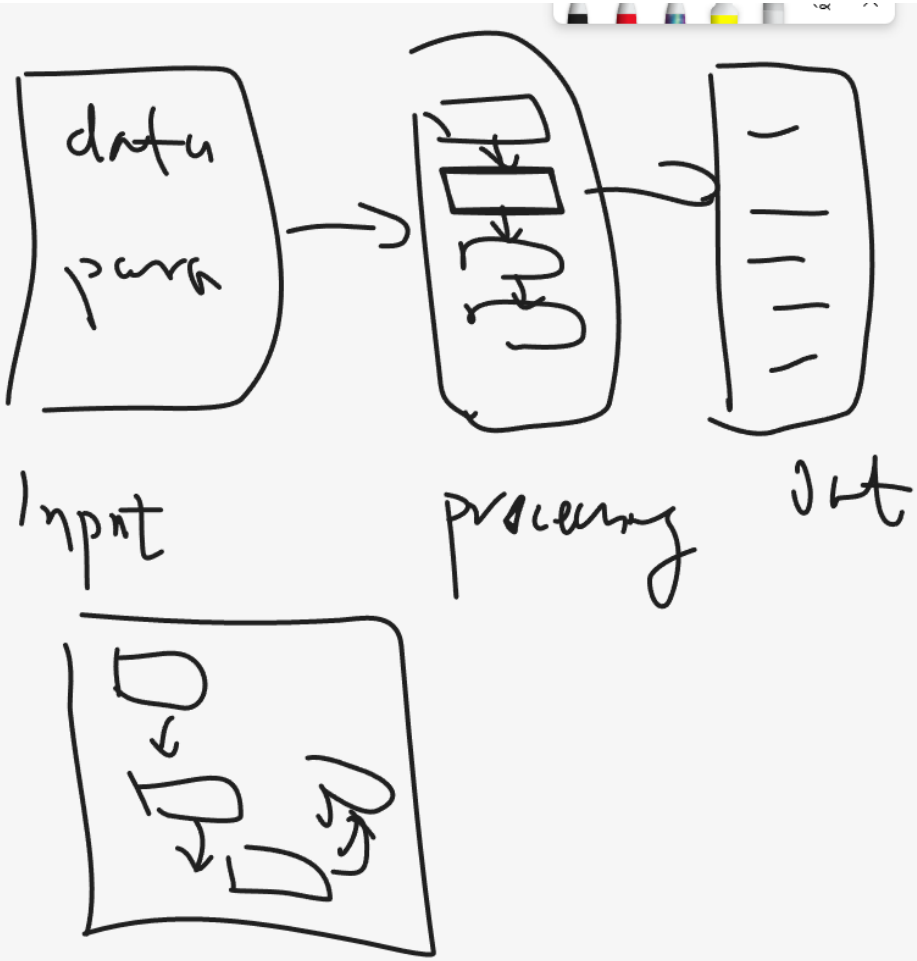
- A system that uses accelerometer and gyroscope data (or a manual override switch) to govern whether or not the suspension should be active
  - Should disable the suspension over smooth ground (accelerometer)
  - Should enable it over bumpy terrain (accelerometer)
  - When climbing hills the system should be in greater favour of disabling the suspension as that's when efficiency is needed most (gyroscope)
- Controls servos to use the
  - Maybe LEDs for feedback



Progress Overview For The Last Week

- ~~Use this template for research notes~~
- ~~What problem does this project attempt to solve?~~
- ~~Why is the research problem important?~~
- A rough idea about how the problem to be solved?
  - Try to survey some related work
  - Go to *Google scholar* and search key words
  - Make some slides to introduce these existing solutions and point out their cons and pros

1. slides: key points, figures, reference format.
2. Survey existing solutions
   2.1. pic
   2.2. architecture, hardware, software
   2.3. overview of the algorithm
   2.4. cons and pros
3. draft your solution
   3.1 architcture, hardware, software
   3.2 rough idea about your algorithm



Problem

- Mountain bikes feature suspension for negotiating rough terrain
  - Wheels move independently to bike
- This causes inefficiency on smooth terrain
  - Some energy wasted compressing the suspension when pedalling
- Most suspension comes with lockout to disable the suspension to solve this problem
  - Lever near the suspension
  - Inconvenient and unsafe to use frequently

Proposed solution

- An embedded system that automatically gauges the terrain currently being traversed and enables/disables suspension accordingly
  - Uses accelerometer/gyro/hall effect data to gauge terrain and rider intention
  - Uses servos to control lockout levers
  - Use LEDs/switch for manual override
  - Uses microcontroller/microprocessor to do the thinking

Research

- Interpreting, filtering, displaying and using sensor data
- Practical/realistic embedded system
- Effectiveness of lockout in the first place

Plan For The Next Two Weeks

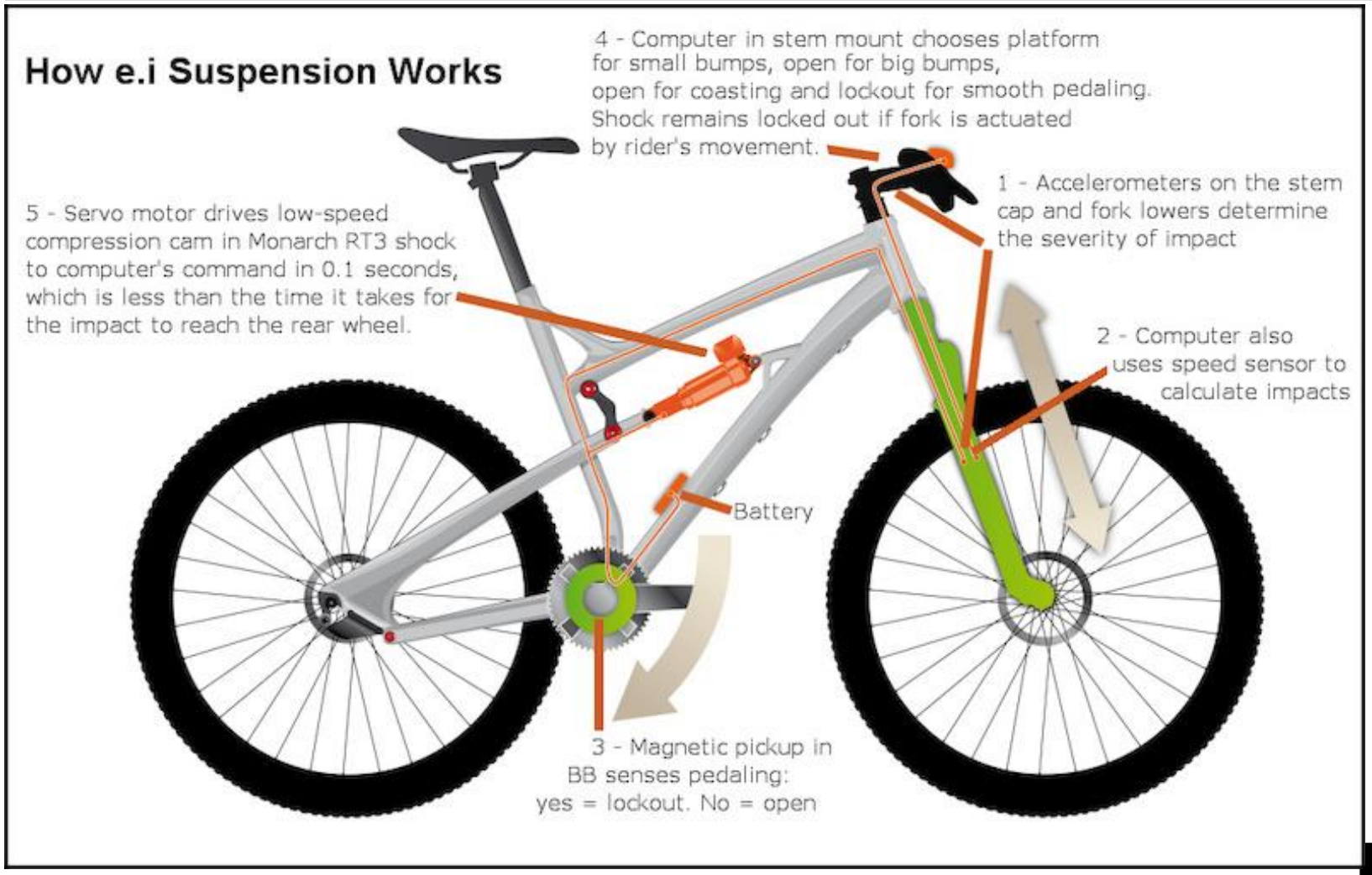
- A rough idea about how the problem to be solved?
  - Try to survey some related work
  - Go to *Google scholar* and search key words
  - Make some slides to introduce these existing solutions and point out their cons and pros
- Task one
- Task two

- Literature review
  - Performance metrics – HOW to compare to existing products
- If all the parts arrive set them up initially
- Parts list

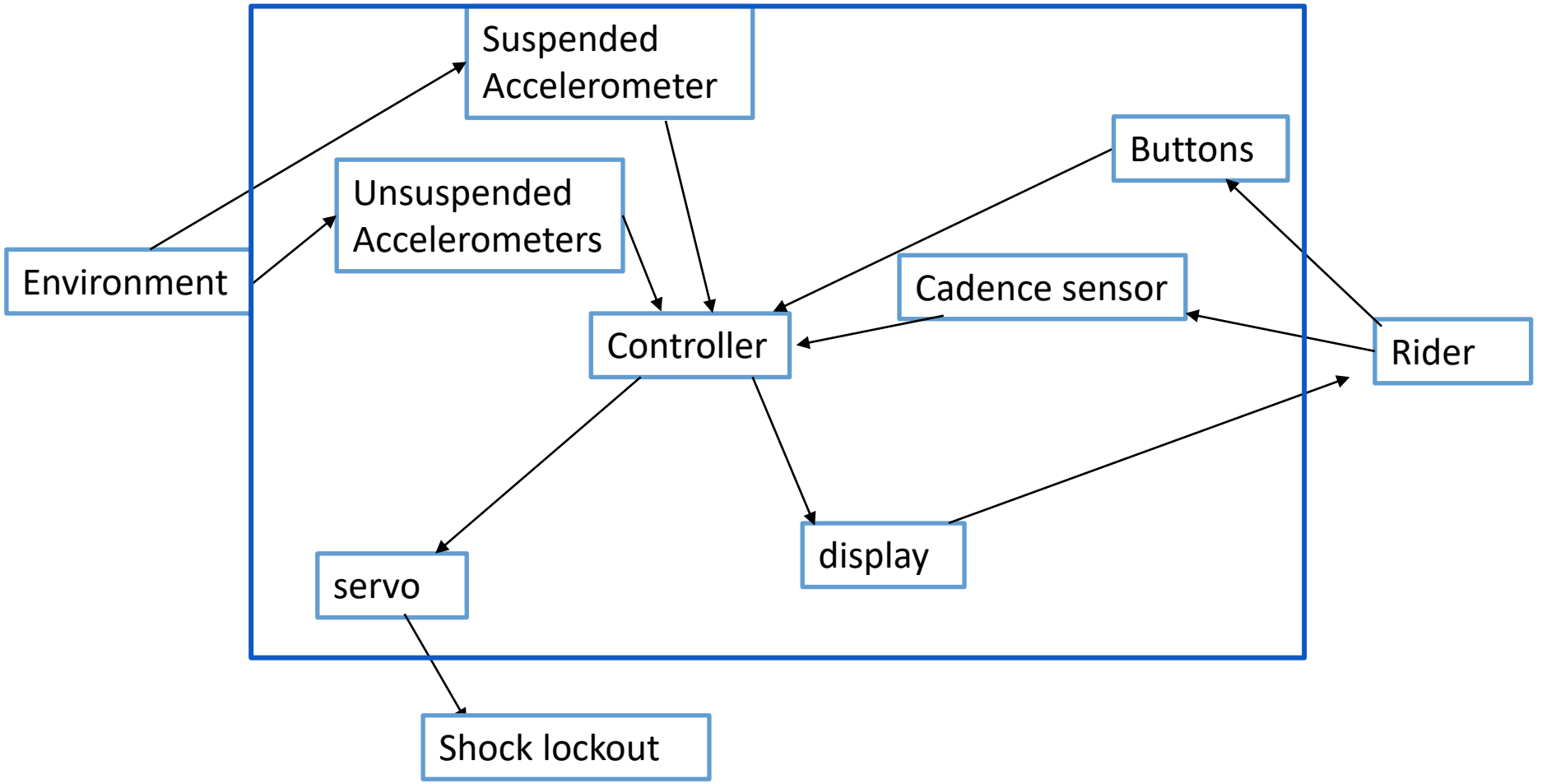
Similar works – Rockshox EI Shock 2011

- Used accelerometer and cadence sensor (hall effect sensor)
  - Accelerometer was mounted at the front
- Controls lockout in rear shock
  - Doesn’t control front fork at all
- Feature cycle computer
- Pros
  - Pretty much none
- Cons
  - Expensive and proprietary
    - Proprietary rear shock, only made for some bikes
  - Only controls rear shock
    - Half the potential energy savings wasted

Pics and architecture



University of East Anglia



Rear shock

servo

Suspended accelerometer

Computer/ controller

Unsuspended accelerometer

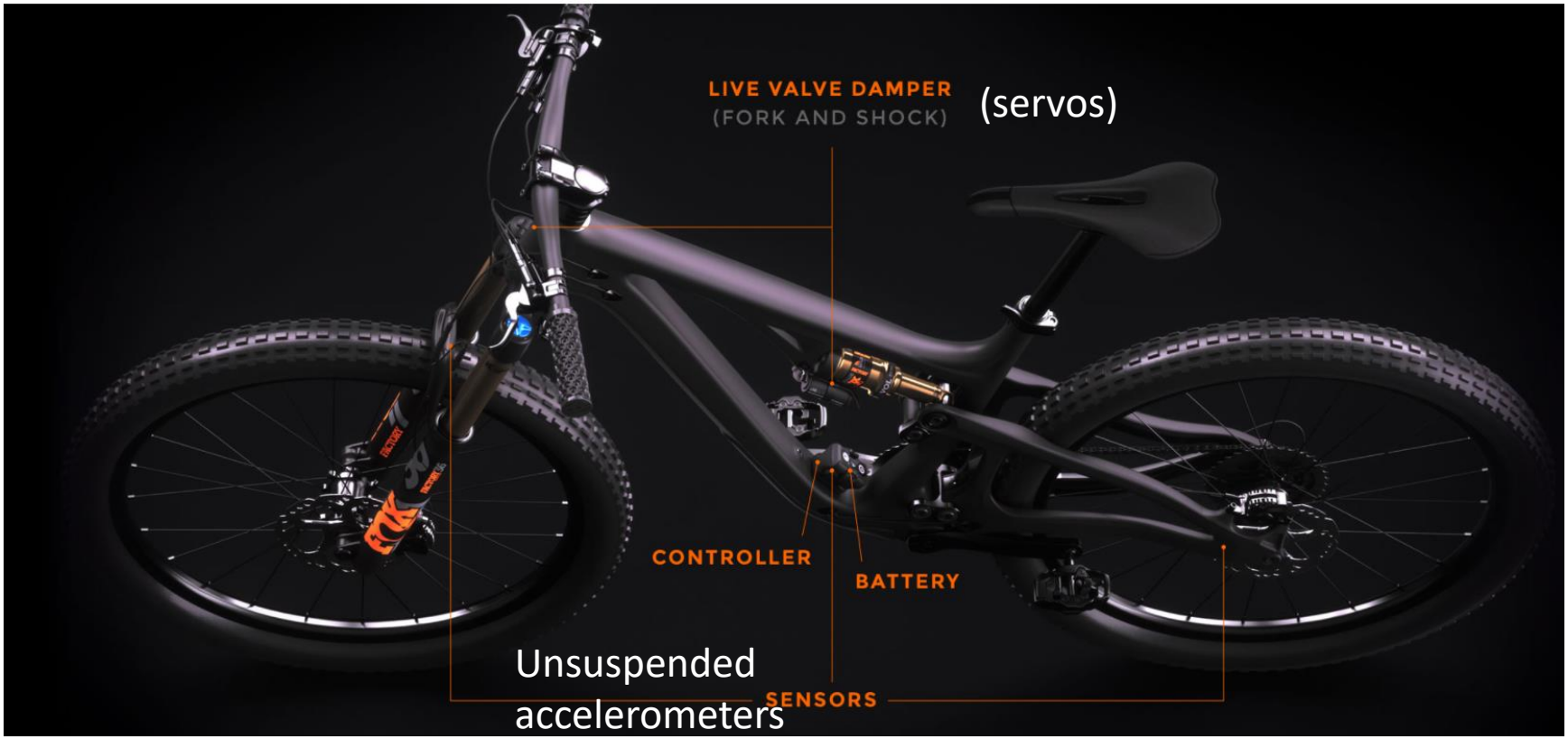
battery

Cadence sensor (hall effect sensors)

Similar works – Fox LiveValve 2019

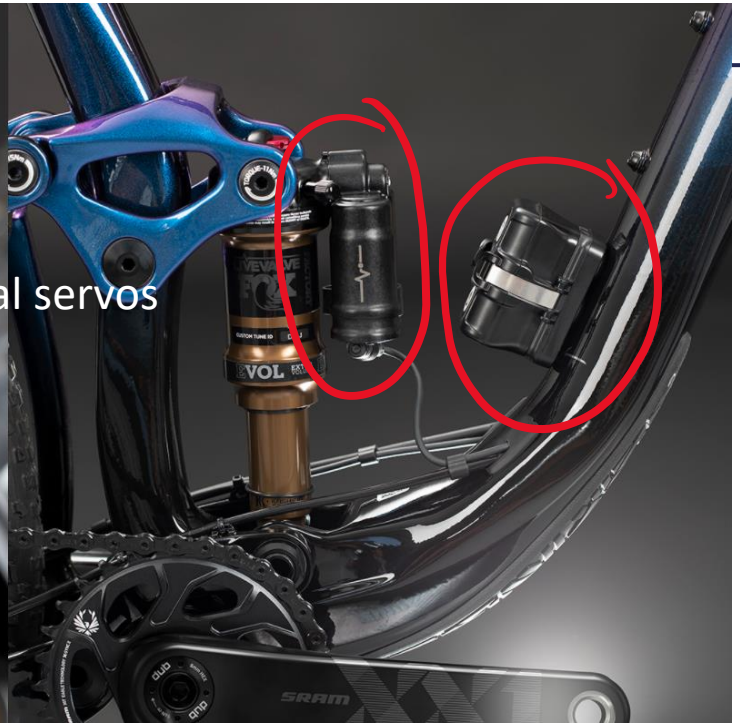
- Uses accelerometers/gyroscope/cadence sensor data
  - Accelerometers mounted next to each wheel
- Controls damping in both front and rear suspension
  - Compression damping instead of binary lockout
  - Uses servos
- Feedback system reacts within 3ms
- Has app/trip computer integration
- Pros
  - Very effective
- Cons
  - Absurdly expensive (3000 pounds)
  - Proprietary suspension and bikes

Pics and architecture



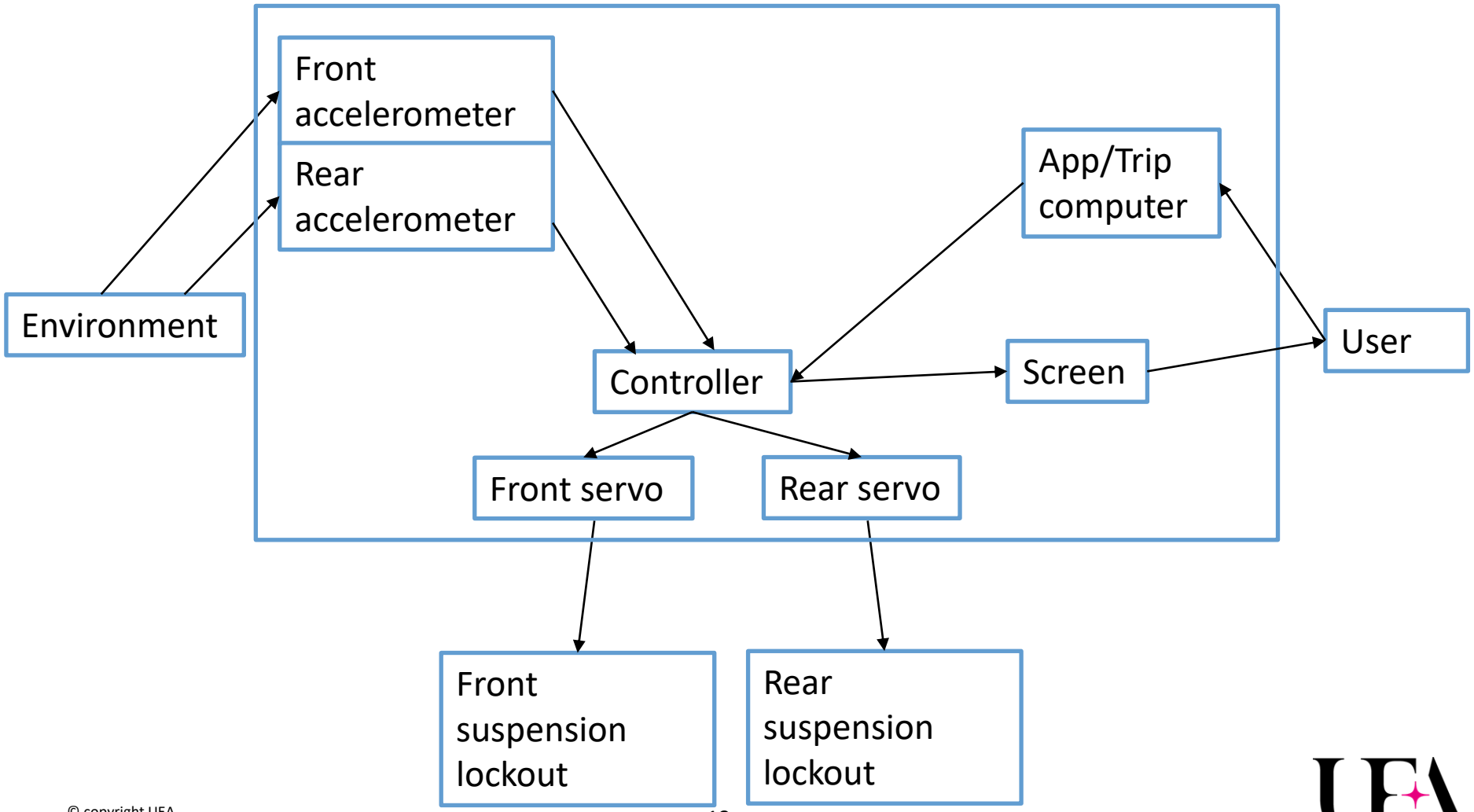
University of East Anglia

Internal servos



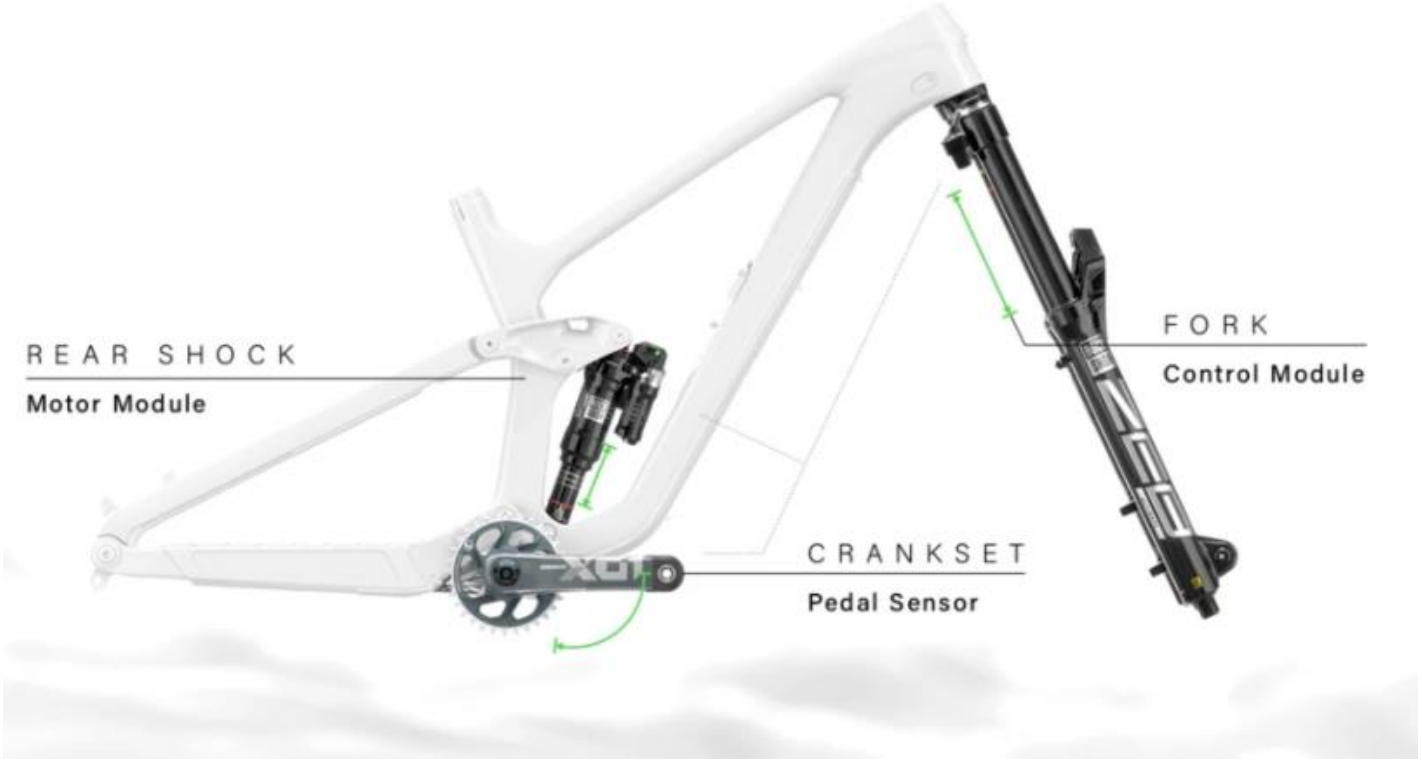
Controller and battery

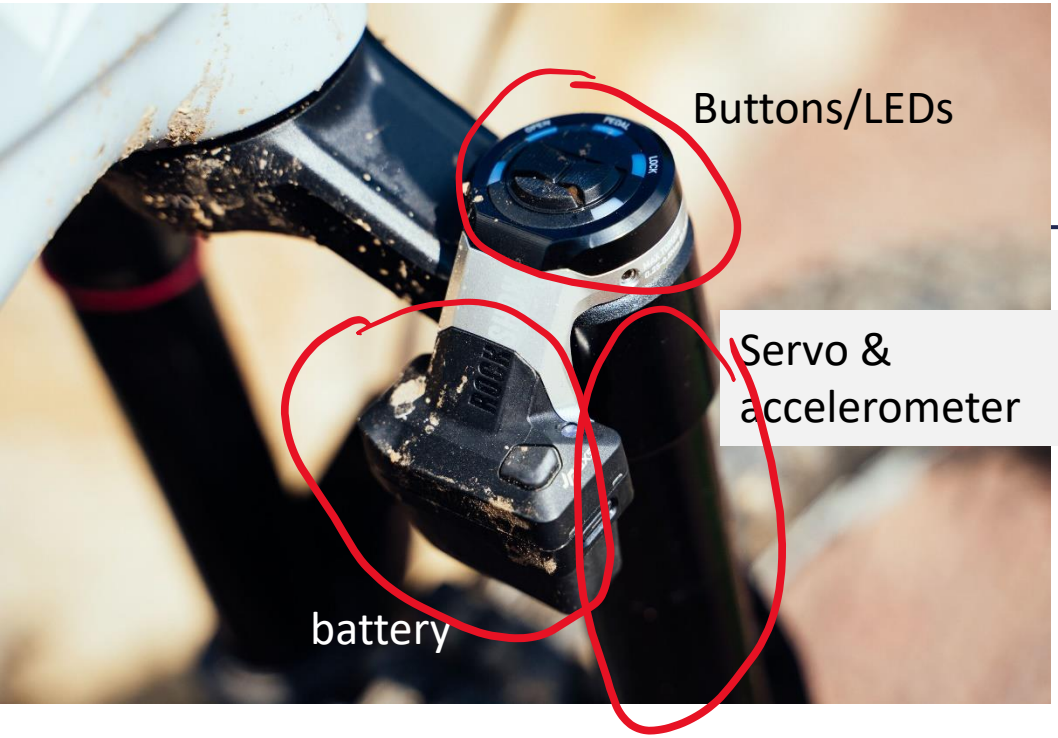
accelerometers

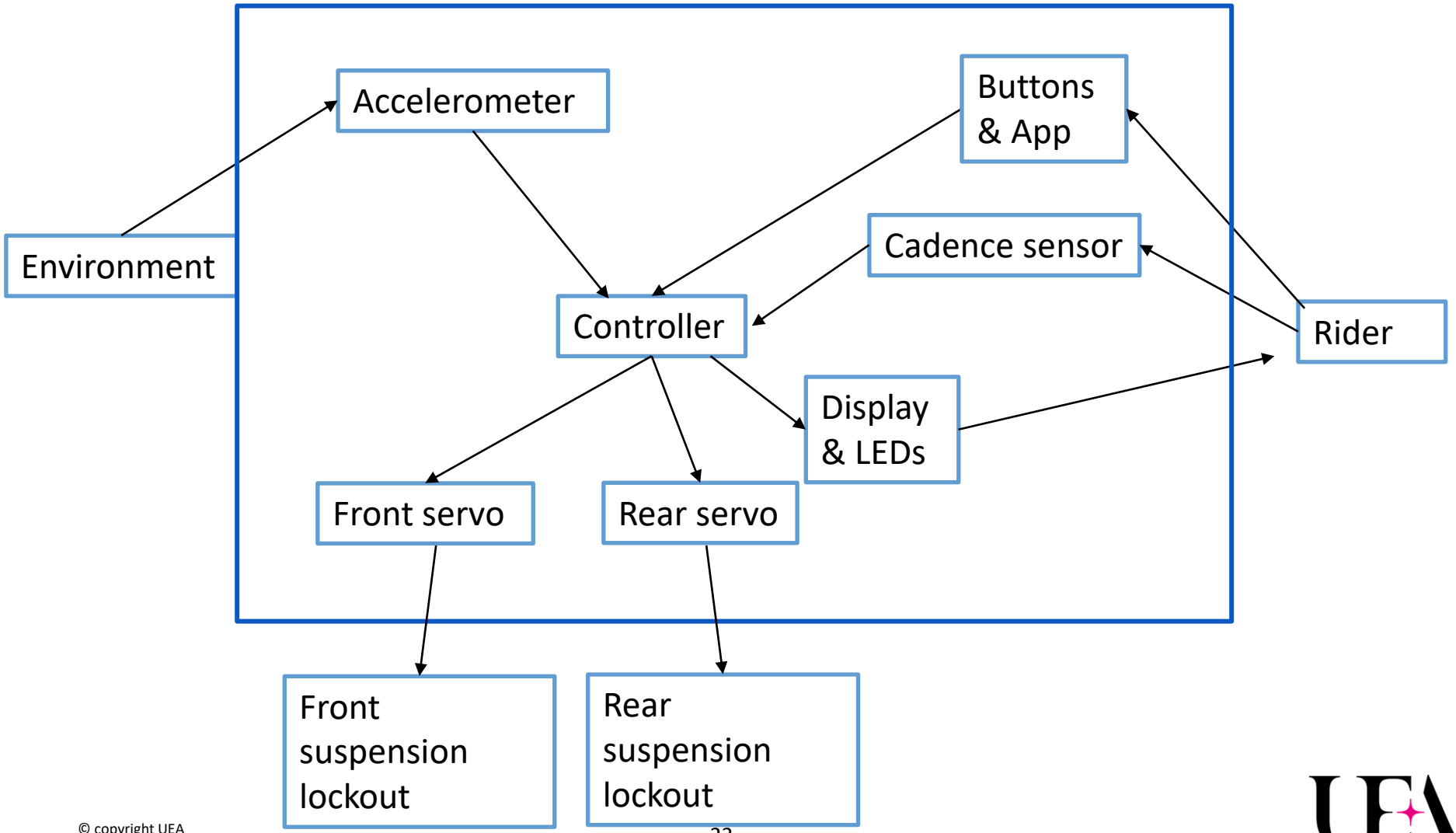


Similar works – Rockshox Flight attendant 2021

- Similar to LiveValve but wireless
  - 5ms response time instead of 3
- Has bias/off set settings for rider configuration, and app integration
- Pros
  - Also works very well and is wireless
- Cons
  - Very expensive and proprietary once again
    - Proprietary suspension and frames





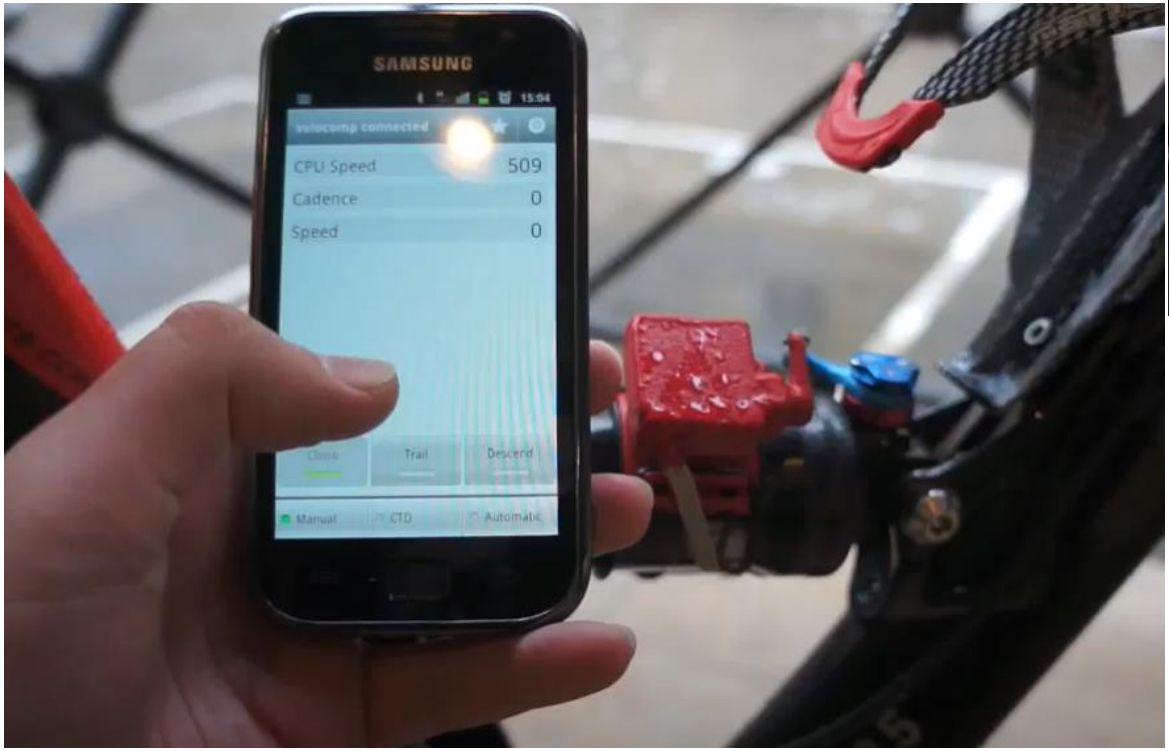


Similar work – Bespoke design by Vladimir Kolotoff 2012

- Uses accelerometers/gyro/hall effect data
- Uses servos to control lockout levers on suspension
  - Would work on a wide variety of suspension
- Features app integration/configuration
  - Control angle threshold, view accelerometer data, etc
- Pros
  - Works on wide variety of suspension/isn't proprietary
  - Inexpensive
- Cons
  - Bespoke/one off project
  - Adjustments made aren't as complex/capable as commercial options.

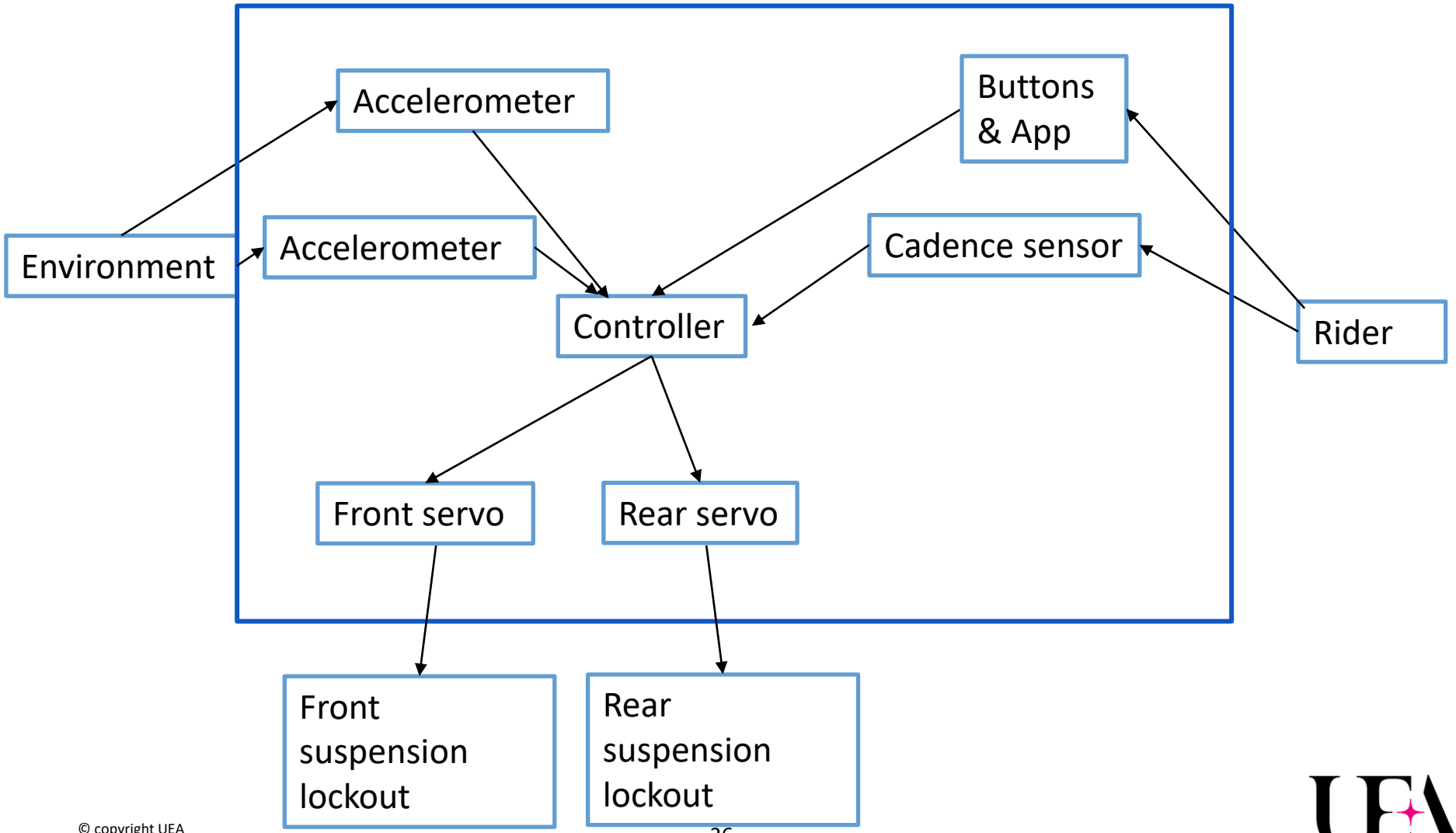

Microcontroller
and battery
(inside the steerer tube)
Three buttons
(under the
rubber handle)
Rear servo
Front servo
Speed sensor
(reed switch)
Cadence sensor
(reed switch,
on the other side)
Accelerometer sensor
(on the fork lowers,
on the other side)

app

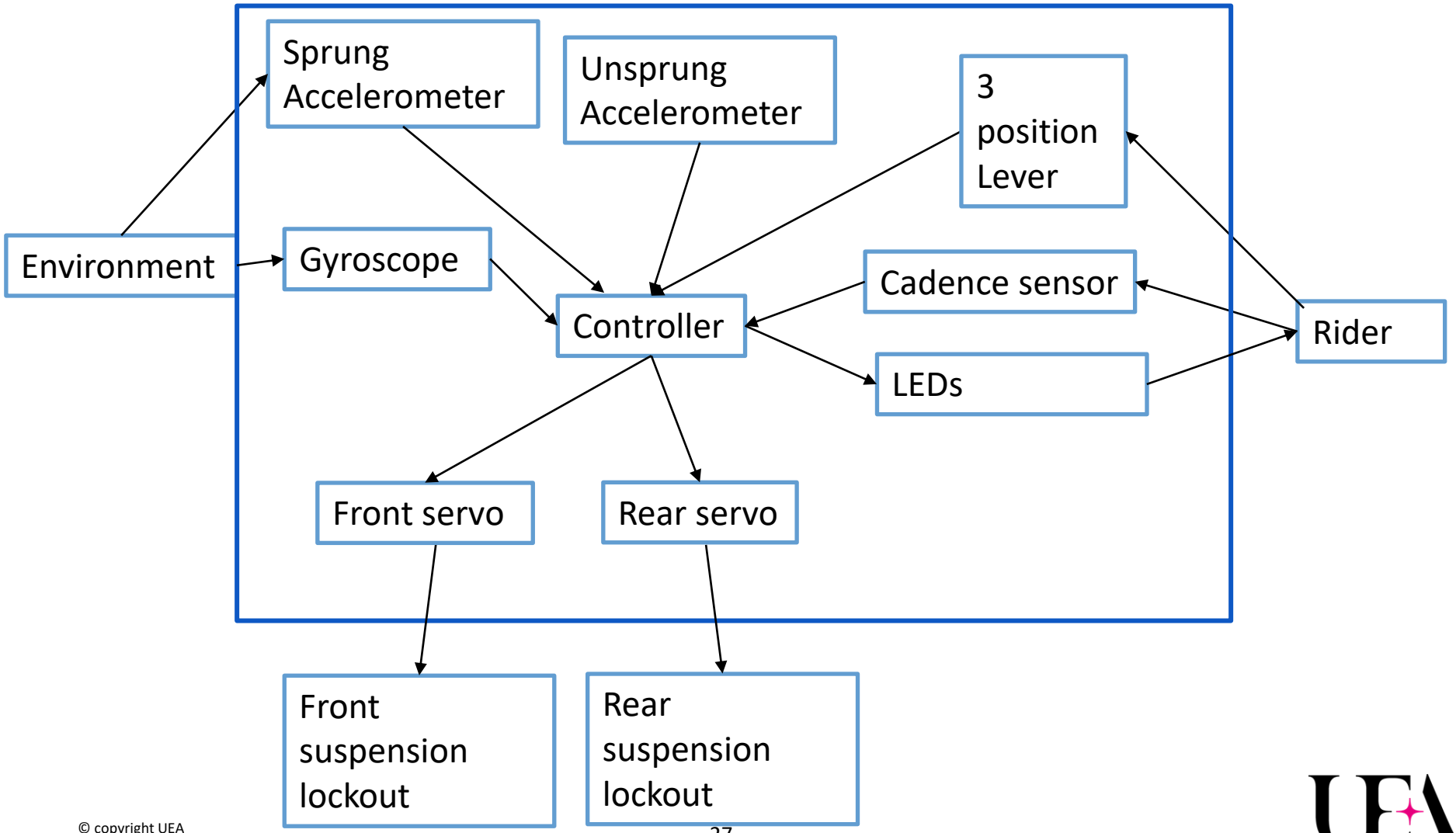


Front and rear servos

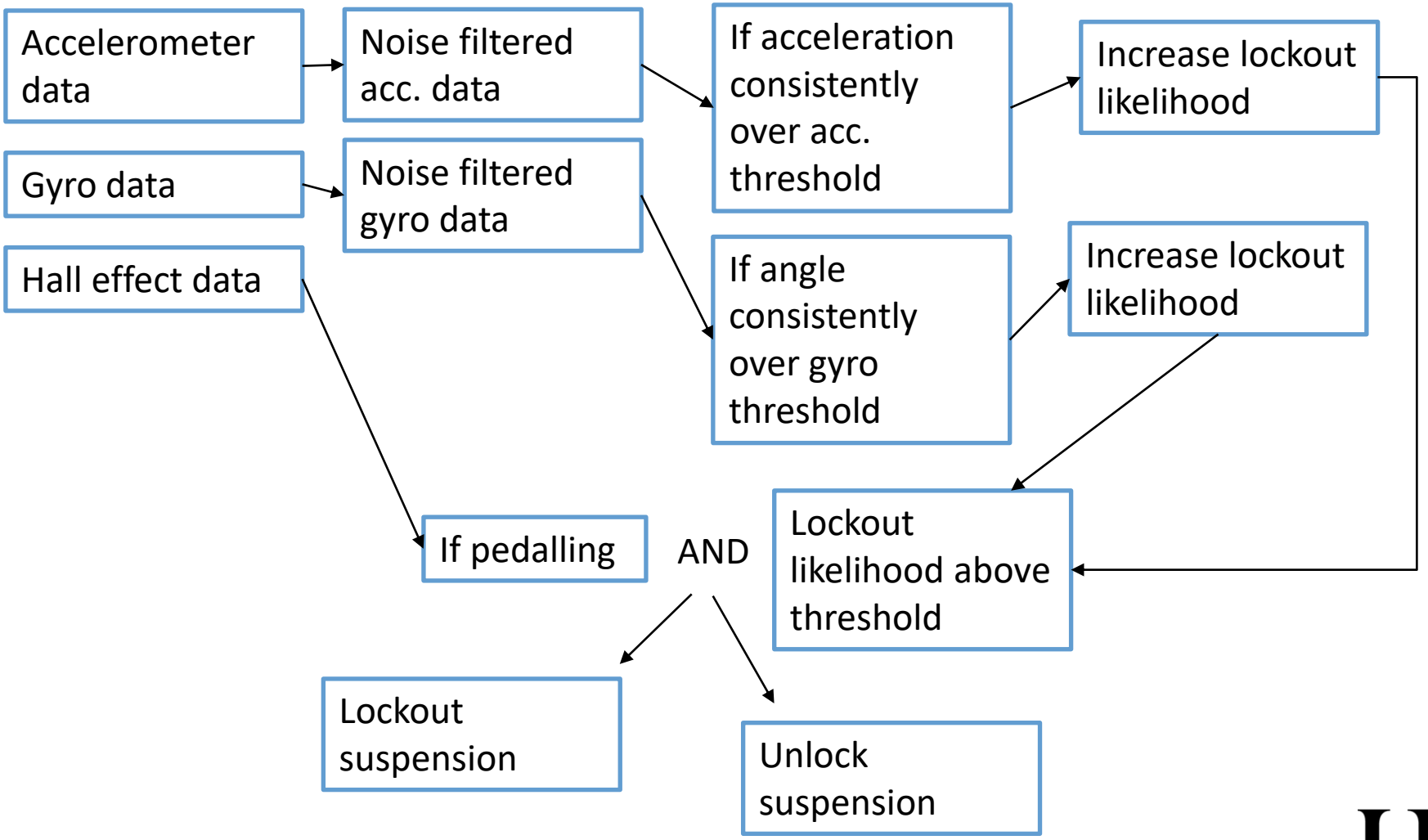
University of East Anglia



My rough architecture



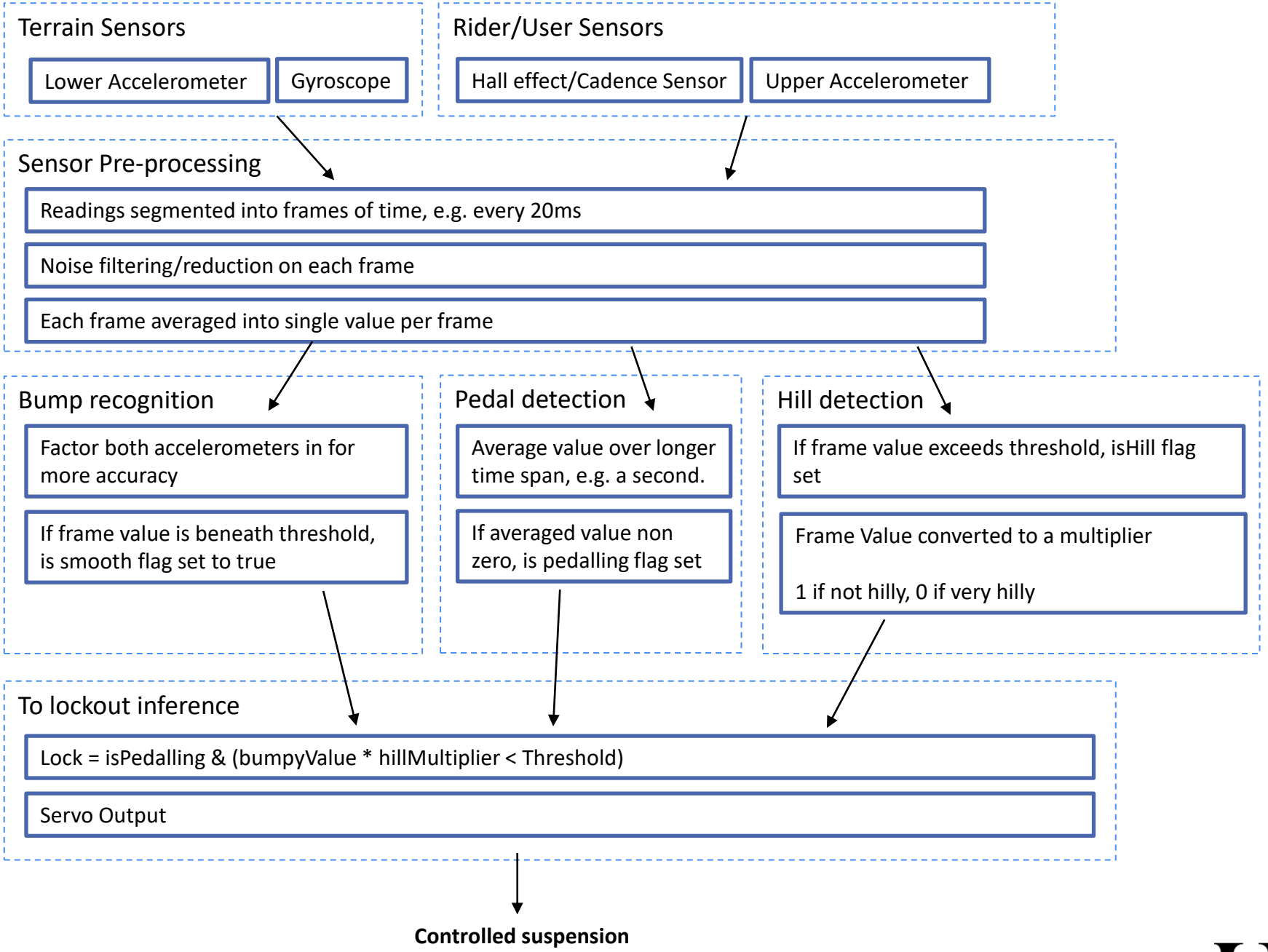
Rough algorithm (of auto mode)

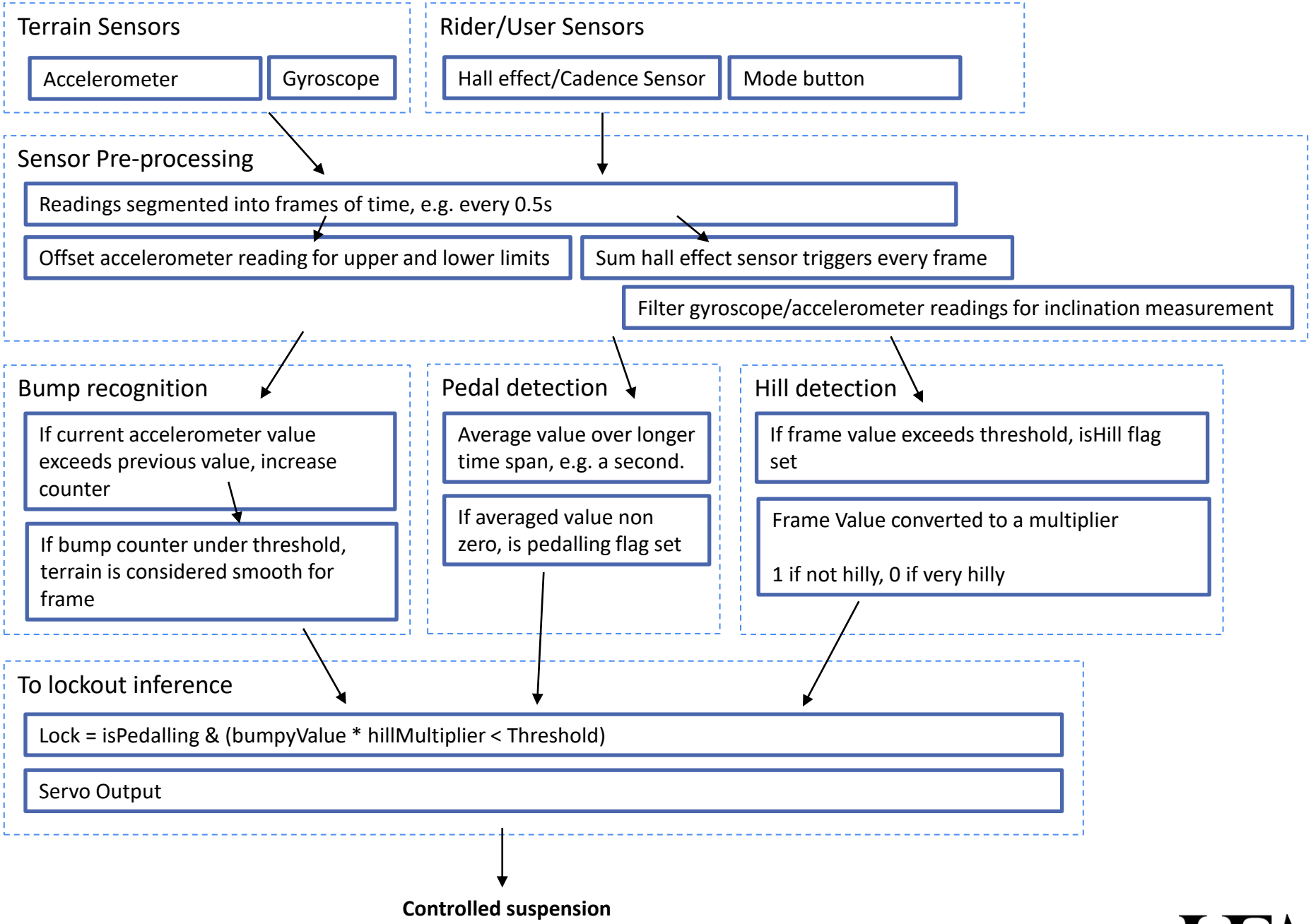


Plan For The Next Week

- Task one
  - Make diagram like page 11
  - Make architecture for system and components
  - Revise IPO input/processing/output model thing
  - Identify behaviors based on input, like pedaling, bumps, etc
  - Algorithm
  - Write literature review!
- Task two
  - Subtask 1
  - Subtask 2
- Task three
  - Subtask 1
  - Subtask 2

System overview diagram



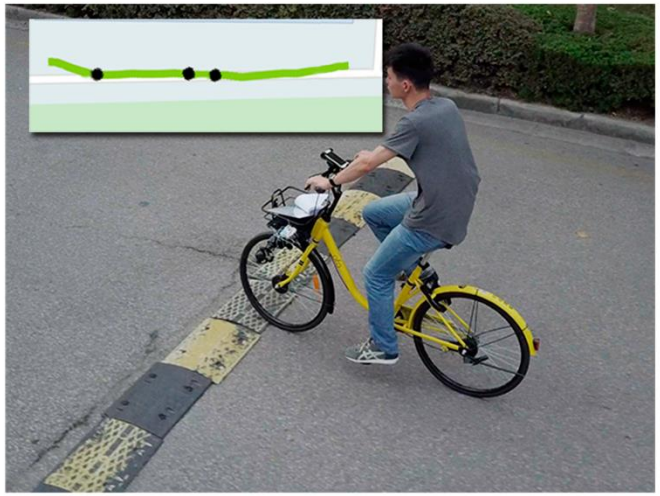


University of East Anglia

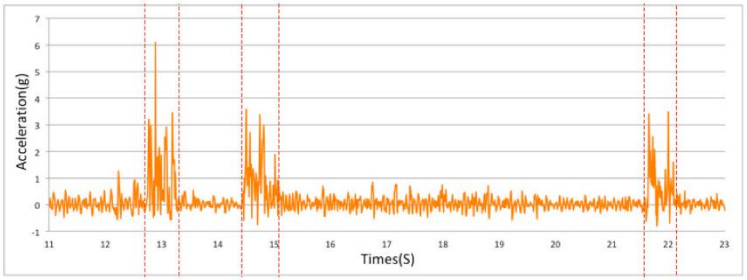
Input	Process	Output
- Hall effect sensor/If pedals are moving - Gyroscope/Angle bike is at - Accelerometers/Acceleration forces	- Remove noise from sensors - Gauge bumpiness of terrain - Gauge inclination of terrain - Gauge if rider is pedalling - If the rider is pedalling over smooth enough and steep enough terrain, disable the suspension for increased efficiency. - Control front and rear servos	- Servos/suspension enabled or disabled

Input behaviours

- Accelerometer – measures acceleration. Bumps will show up as spikes on graph, or jolts of acceleration
  - Sprung accelerometer will be smoothed out due to bumps being smoothed by suspension
  - Can classify peaks exceeding threshold as bumps
- Gyroscope – measures inclination. Simply measures angle
- Hall effect sensor – measures pedalling. If the sensor is triggered on and off regularly, pedals are being rotated regularly and rider is pedalling.
  - Can use on/off rate exceeding a threshold to measure if pedalling or not



(a)



(b)

University of East Anglia

Intro output behaviours/why

Things for next time

- Write progress report
- Get pi up and running
  - Dabble with components
    - Servos, accelerometer, etc

Progress Overview

Main Tasks	Starting Date	Deadline	Progress
Course on Pattern Discovery	2016/11/7	2016/12/11	3 modules out of 4
Programming in Java	2016/11/4	2016/11/30	70%
Android Programming	2016/11/25	2016/12/25	0%

Milestones

Progress Report

25-30 Jan. 2018

For example: Adam Cook
University of East Anglia

31-Jan-18

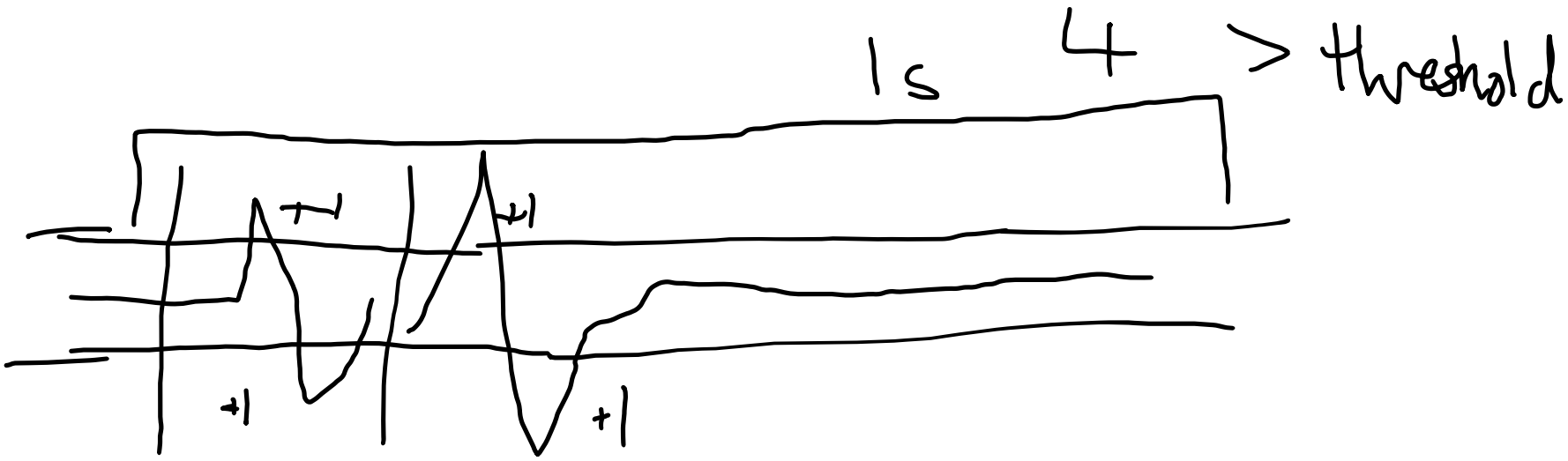
Progress For The Last Week

- Got accelerometer/gyro/hall effect working
- Logs to file
- Averages out values over time
  - Doesn't work for acceleration since bumps are both + and – G so they cancel out
- Plan for next week
  - Get some data!

Progress For The Last Week

- Got servos working
  - They were continuous for some reason
    - Ordered 180 degree ones
  - Worked on loop
    - If acceleration goes beyond or below threshold add “to isBumpy”
    - Averaged out over a longer period
      - If that average is beneath a threshold && cadence above cadence threshold && angle above angle
      - Threshold lock the suspension
      - Otherwise unlock
- Milestone: Components operational

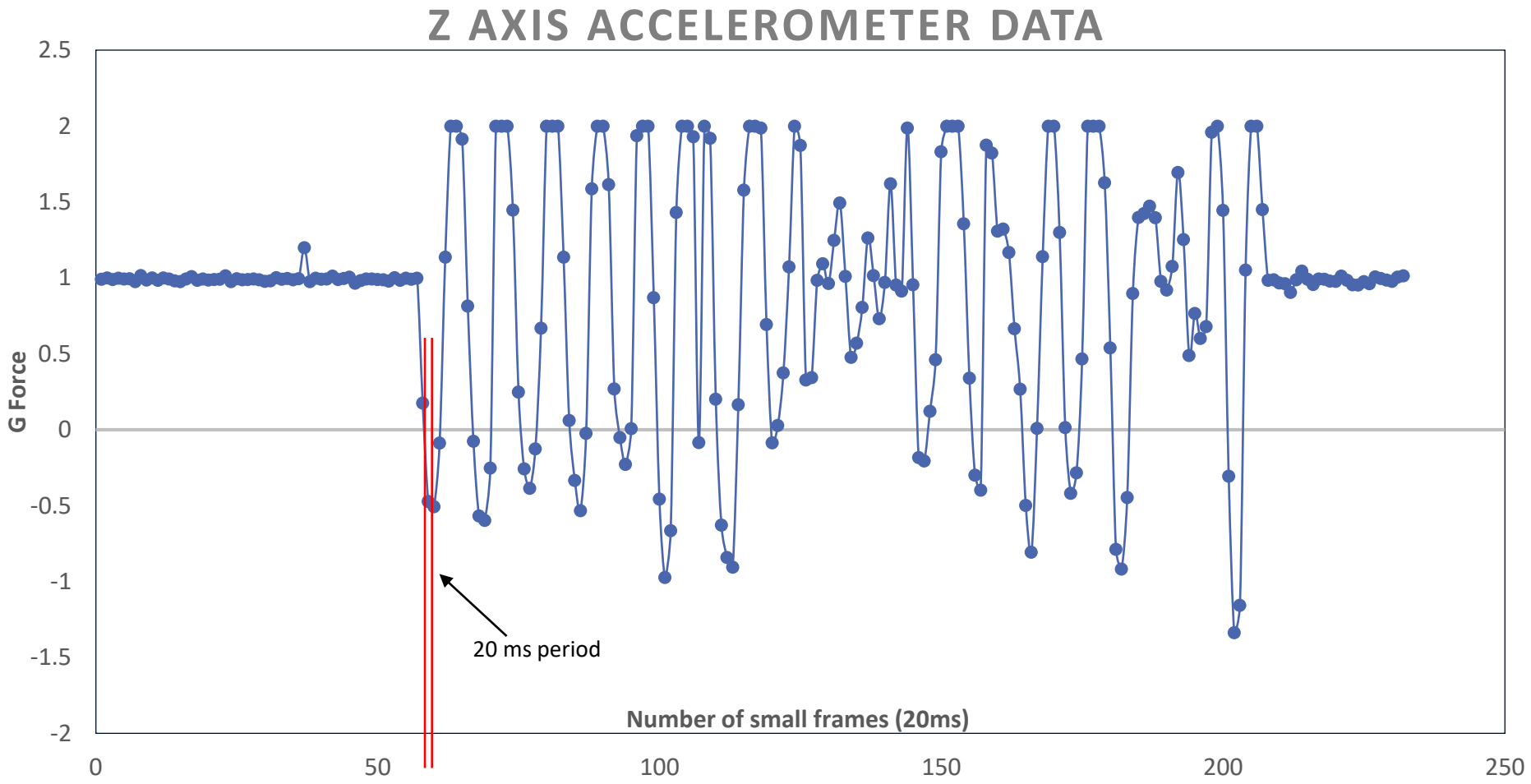
Binarizing prototype



Progress For The Last Week (cont.)

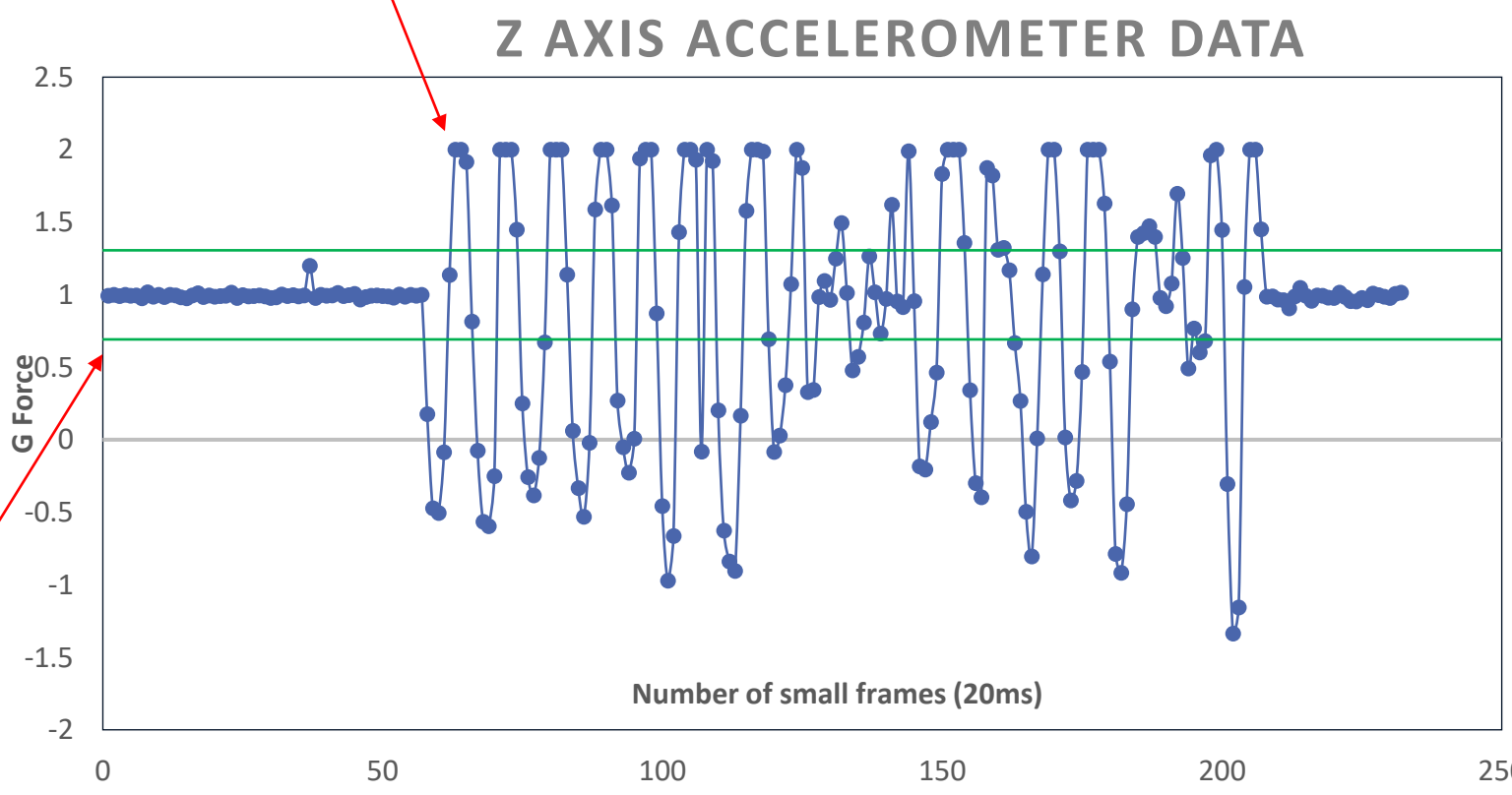
- Task two…
- Sliding window
- Get pics
  - Raw data
  - Process of binarization

Feature extraction/binarization of bumps



- Each dot is a measurement every 20ms
- An upper and lower limit is chosen
  - This is an offset of previous readings
    - Eg, past readings averaged over a second +- 2m/s/s
    - Can't be a fixed limit as value can change overtime – eg an inclination
  - If value exceeds this, increment a bump counter

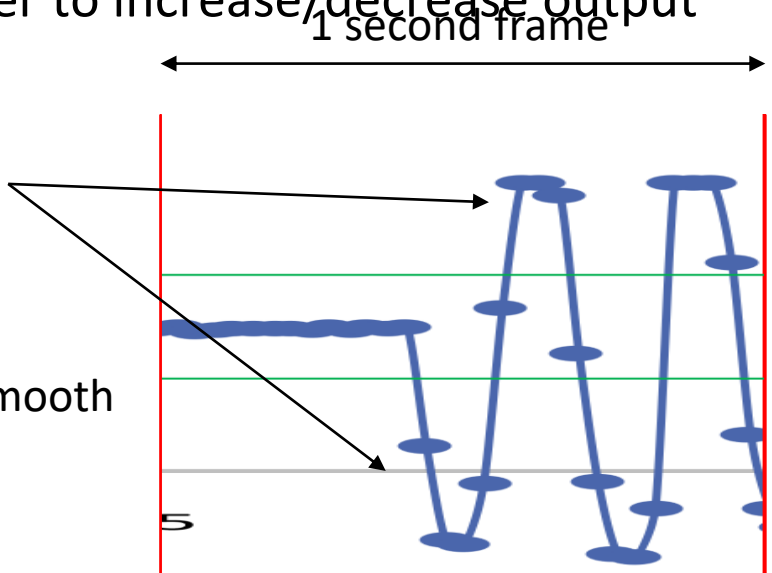
Reading exceeds limit - bump



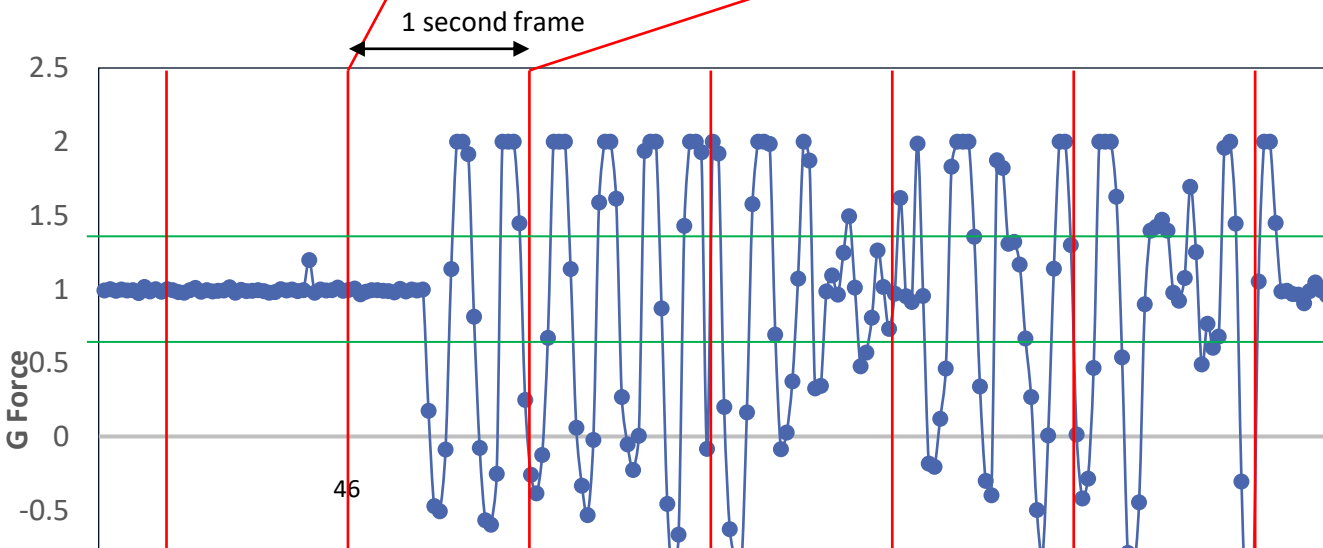
Example limit

- Over the course of a longer set period/frame eg 1 second, the counter is checked against a threshold
- This frame can be shorter or longer to increase/decrease output responsiveness

Limits exceeded 16 times – 16 bumps in the past second



- Value compared with threshold
  - Eg. If 16 < threshold, terrain is smooth
    - Vice versa
    - Terrain is binarized!
- Counter is reset to 0 for next frame



Feature extraction for pedalling and hills

- Same is done for the hall effect sensor
  - Every time the sensor is triggered a pedal stroke has occurred
  - Over frame (1 second) the total count is measured and compared
    - Eg. 43 > cadenceThreshold
    - Reset to 0 for next frame
- Current angle can be taken every second

Final output

- Every frame (1 second) a decision is made and servo state is changed/maintained
- This time frame can be shorter or longer to increase/decrease output responsiveness

```
if bumpCount < bumpThreshold and hallEffectCount >= cadenceThreshold and y_rotation > angleThreshold:
```

- Suspension with be disabled if (higher pedalling efficiency):
  - Terrain is smooth enough AND
  - Rider is pedalling AND
  - Rider is on flat or uphill (Will change to be a likelihood multiplier)
- Otherwise suspension will remain active/enabled
  - To be on the safe side

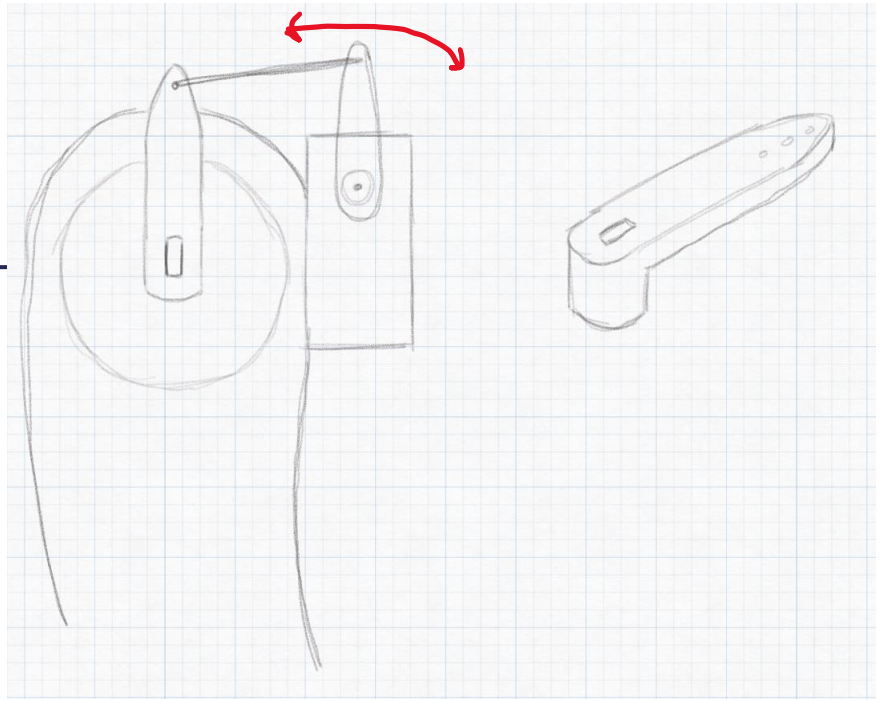
For next time

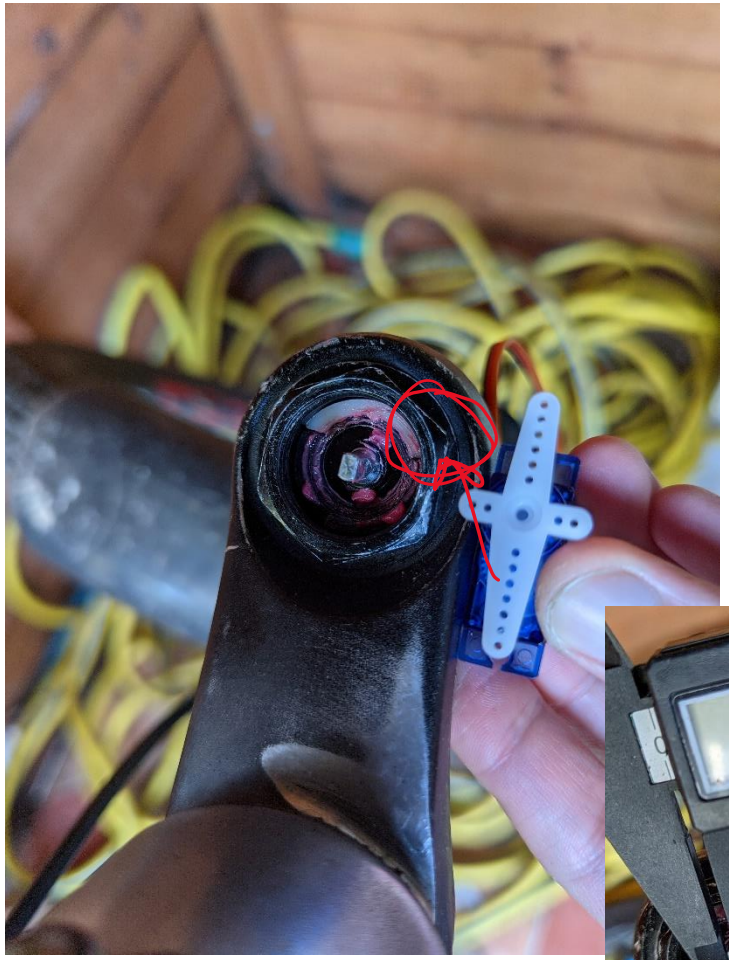
- GET REAL WORLD DATA
  - Find a way to attach to bike
  - Come up with values for thresholds based off of said data
- Maybe look into expanding the system
  - Override switch
  - Status LED/screen
  - Research into parts for a “production model”
    - AKA not using a raspberry pi hooked up to a powerbank

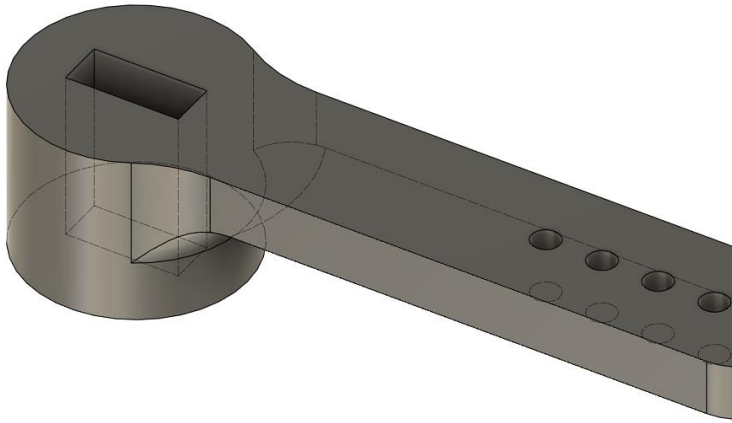
Progress for last week

- Real world data? Nope
  - Power bank died 🙃
- Did other stuff instead
  - Got LED working
  - Got code for button to work
    - Will switch modes
      - Auto, Manual lockout, manual open
      - Led will go green, red and blue respectively
  - Designed lever for lockout on front shock

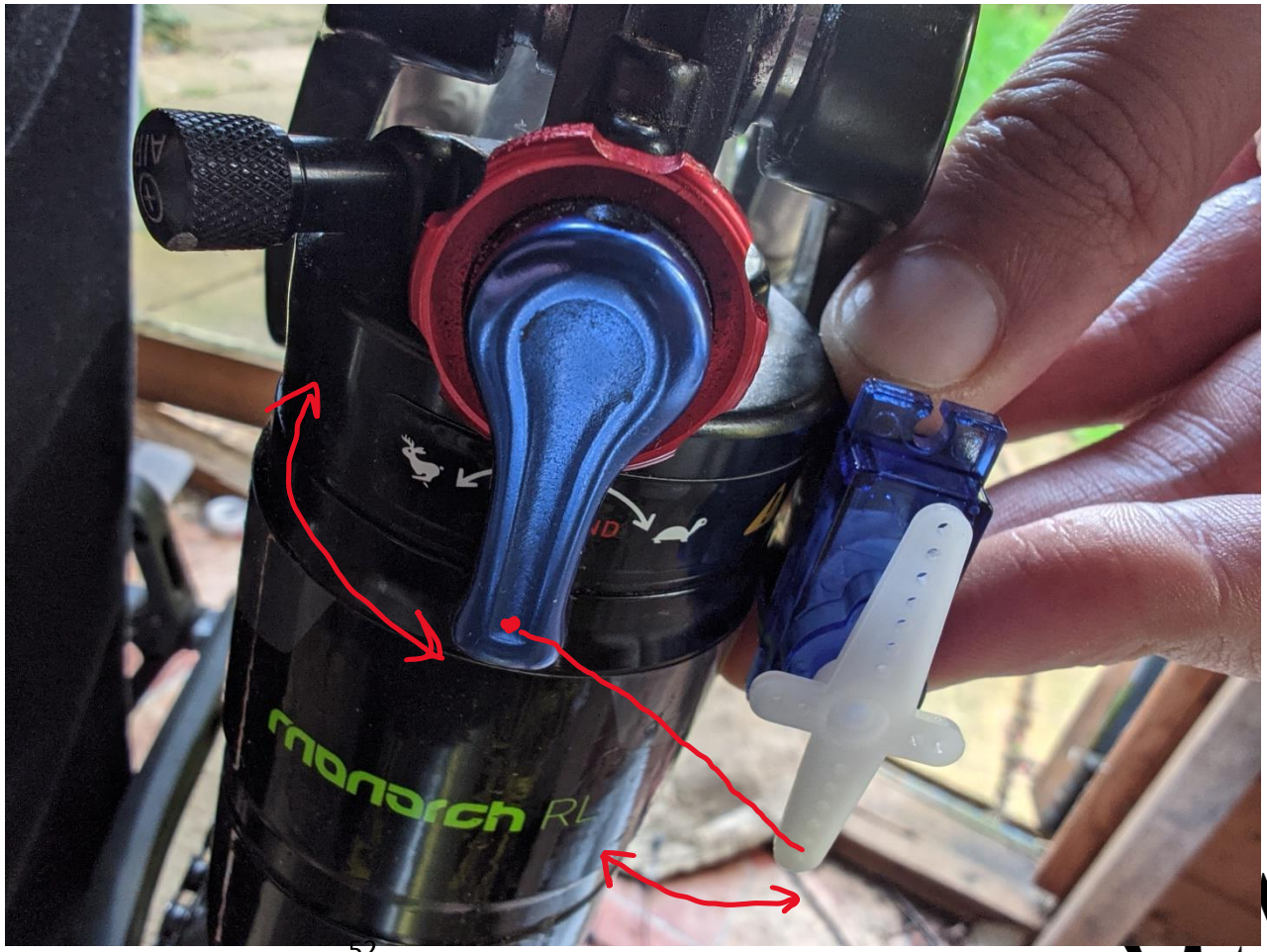
Front suspension lever



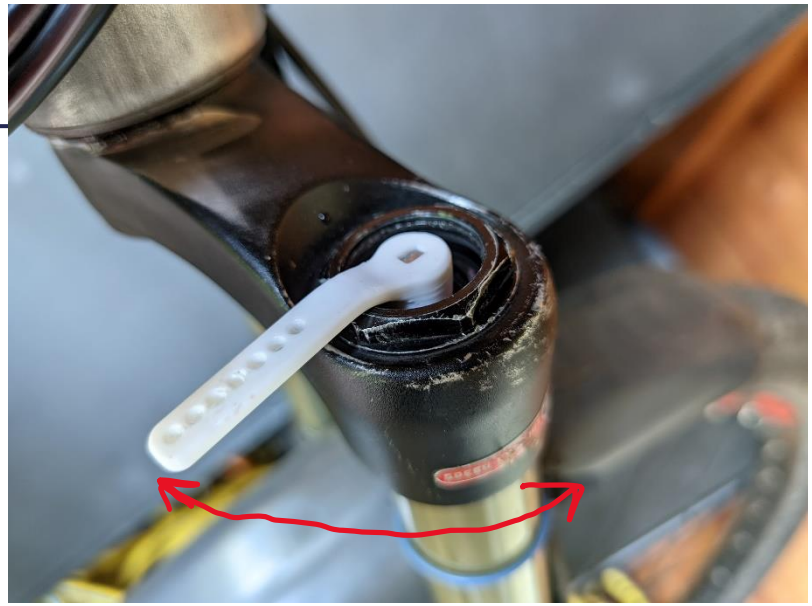




- Will drill a hole in pre existing lever



Progress for last week



- Power bank still absent
- Did other stuff instead
  - Printed lever for front suspension
    - Thanks Ben!
  - Worked on code a bit
    - Changed angle input from binary (is it downhill or uphill) to a multiplier
    - Angle ranges from -90 to 90
      - If -90 (vertical downhill) multiplier is 0
      - If 90 (vertical uphill) multiplier is 2
      - If 0 (flat) multiplier is 1
      - Multiplier multiplies bump threshold
        - The harder it is to pedal the more it favours locking the suspension

Next week/holiday

- Finish everything
  - Solder button
  - Maybe make system react to bumps instantly as well as over time
  - Hook up system to bike
  - Hopefully get power bank and get some data
  - Modify thresholds accordingly
  - Call it a day

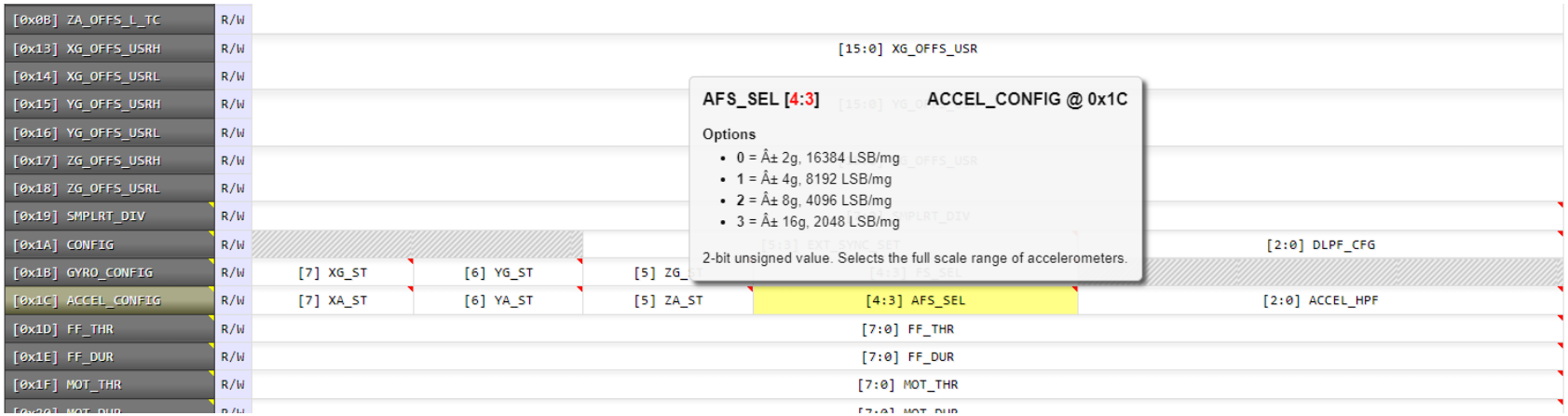
20-April-22 Easter break

Progress for last week

- Soldered button
- Fixed bike
- Power bank arrived
- Wrote separate logging program
  - Uses button to start/stop recording and led for status
- Data results sucked
  - G range was too small, only +-2g
  - Files were out of order
    - Named by date/time but pi's date/time was random for some reason
- Needed to set a control register to set the g range higher

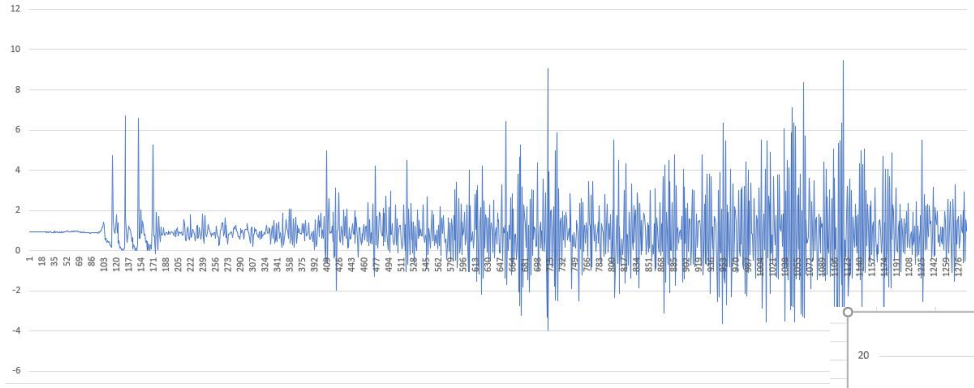
Setting the control register

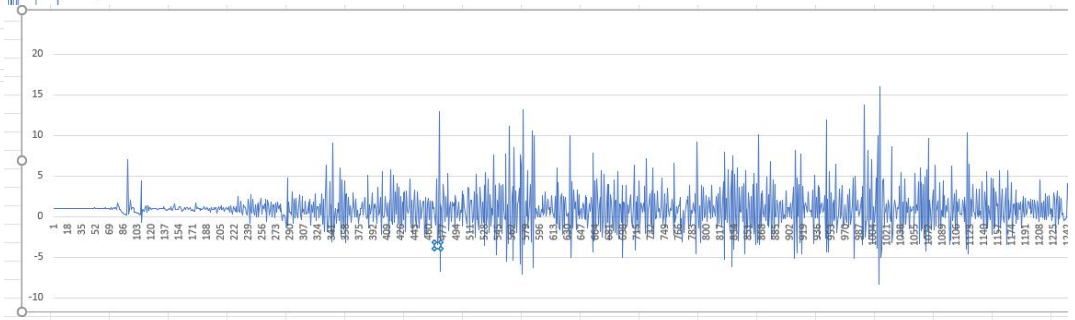
- Looked up the register map for address
  - Bits 3 and 4 set range of accelerometer
    - 0 for 2g, 1 for 4g, 2 for 8g, 3 for 16g
- Had to set address to “24”
  - This is 00011000 in binary
    - Sets bits 3 and 4 to 1 or 3



values

- Will use un suspended position
  - Both similar but unsuspended will be more consistent since suspended will be affected by suspension lockout
  - You can see the suspension working!
    - Seems to affect bigger bumps more, smaller peaks but similar elsewhere





Defining a bump

- Got data from
  - Smooth ground
  - Bumpy ground
  - Inbetween gravel
  - Commute to and from uni (a mix but mostly should be considered smooth)
  - From all this I consider a bump roughly a change of 3 - 5 g
    - Smooth ground doesn’t have any of these
    - Bumpy has a lot of these
    - Inbetween has a fair amount
    - Commute has a rare amount

Bump frequency

- Bumps have been defined
  - Frequency will now determine terrain type
  - 64 bumps over 100 seconds gained from data
    - 0.64 bumps a second
    - On the very conservative side – the suspension should favour being active
  - 6643 - 2755 = 3888
  - 438 seconds for 17172 samples
  - Will need to obviously be experimented with!
  - Milestone: Data aquired!

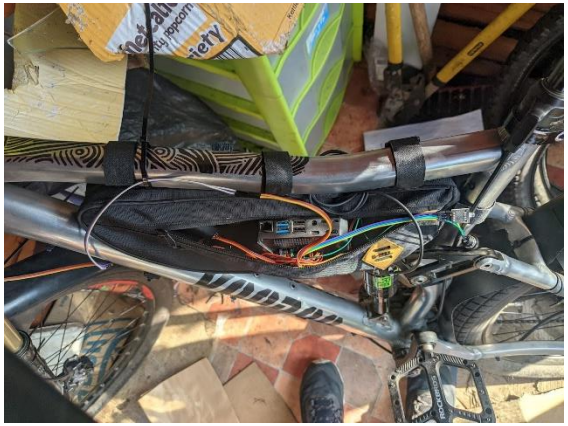
New issue

- Turns out calculating tilt from the accelerometer/gyroscope doesn't work when there's vibrations
  - Makes sense, can't tell what's gravity or a bump
  - I thought I could average the value out but didn't work
- Guess I'll have to ditch the steepness multiplier
  - Alternatives for future
    - Power meter – more power put into pedals more likely to lockout?
    - Speed sensor – the slower you're going the more likely to lockout?
    - (VERY EXPENSIVE) vibration tolerant tilt sensor
    - Could combine both sources and crosscheck
      - No time though

Things for next time

- Hook up servos and hall effect sensor to bike
- Give it a test run and refine values!
- Write dreaded report

Pics of mounting everything



Milestone: System complete!

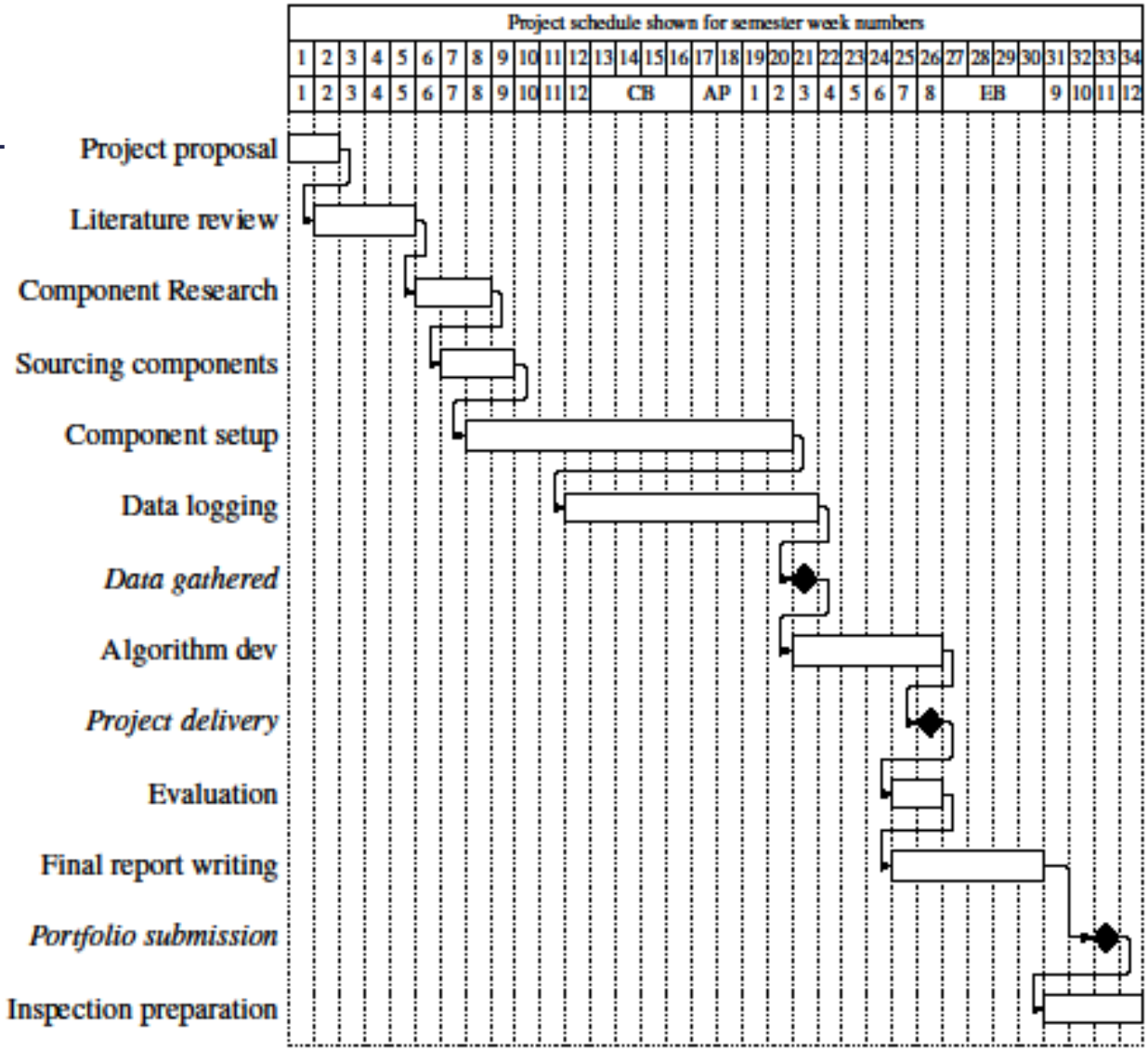
- Changes suspension by itself over rough terrain
- Opens suspension if not pedalling
- Locks suspension over smooth terrain
- Button and status LED work

- Video on teams

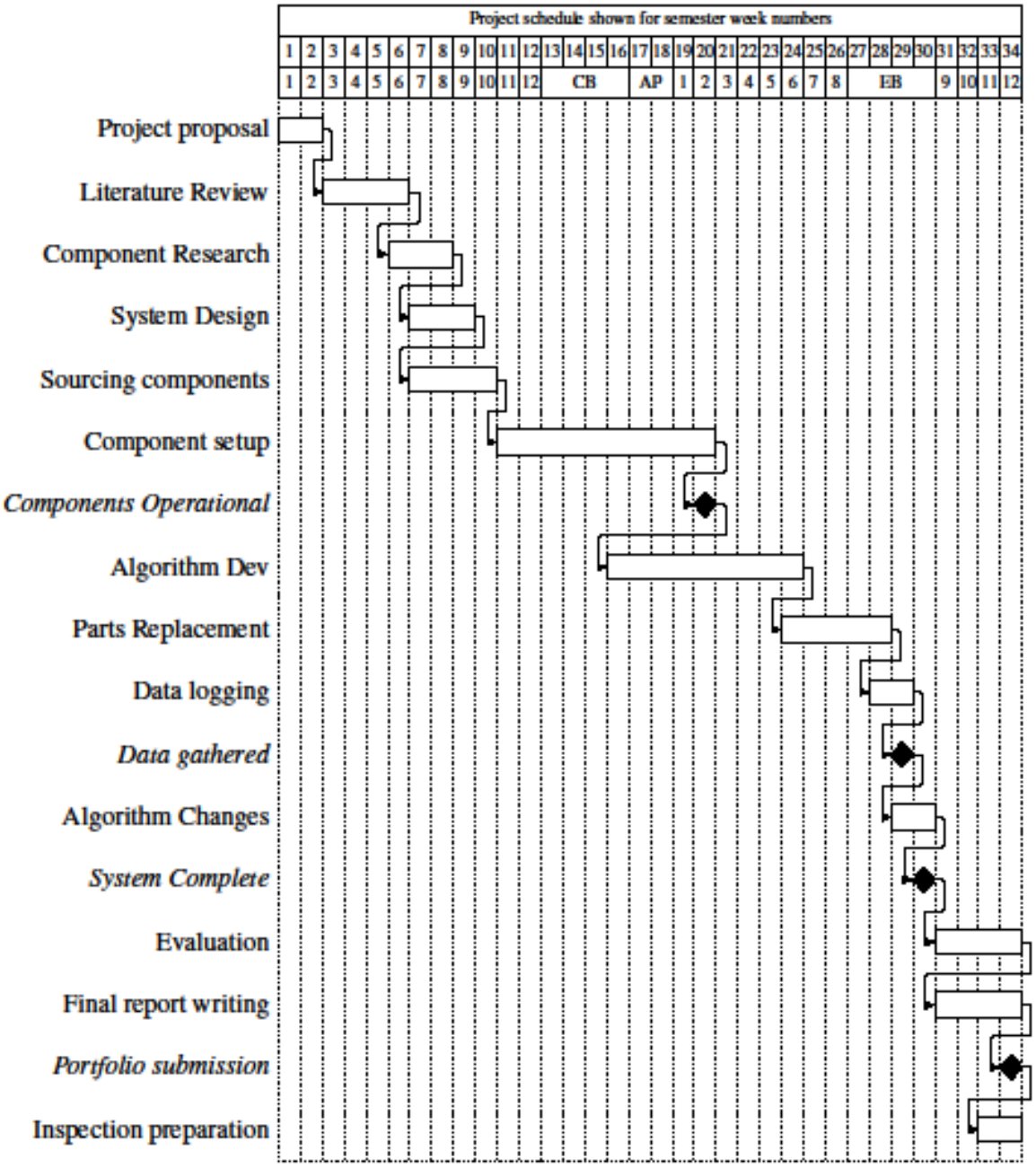
For next time

- Write report
  - Redo diagrams
  - Make graphs
  - Write a whole lot

Initial Gantt



Final Gantt



Progress Overview

Main Tasks	Starting Date	Deadline	Progress
Course on Pattern Discovery	2016/11/7	2016/12/11	3 modules out of 4
Programming in Java	2016/11/4	2016/11/30	70%
Android Programming	2016/11/25	2016/12/25	0%