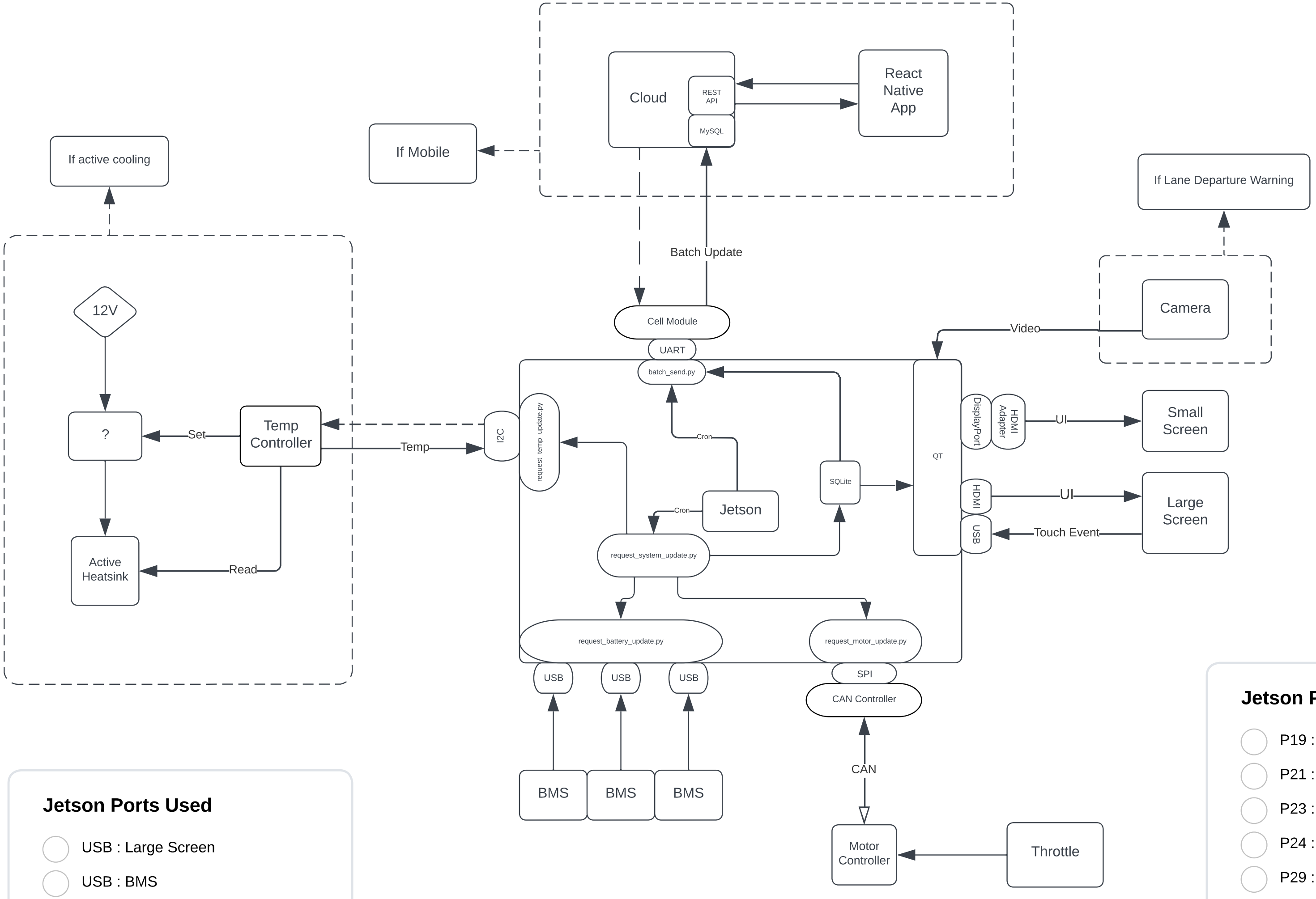
Custom serial protocol between jetson and temp controller



Jetson Ports Used

○ USB : Large Screen

○ USB : BMS

○ USB : BMS

○ USB : BMS

○ HDMI : Large Screen

○ DisplayPort : Small Screen

● Untitled color item

Jetson Pins Used

○ P19 : CAN SPI Data Input

○ P21 : CAN SPI Data Output

○ P23 : CAN SPI Clock Input

○ P24 : CAN SPI Chip Select

○ P29 : CAN SPI Interrupt

○ P8 : CELL UART TX

○ P10 : CELL UART RX

○ P36 : CELL UART CTS

○ P27 : ESP I2C SDA

○ P28 : ESP I2C SCL

system_scripts

November 4, 2022

0.0.1 request_system_update

- request_battery_update
- request_motor_update
- request_temp_update

request_battery_update

[1]:

```
import dalybms

# bms_ports = ["/dev/ttyUSB0", "/dev/ttyUSB1", "/dev/ttyUSB2"]

# responses = []

# for bms in bms_ports:
#     dalybms.connect(bms)
#     response_dict = dalybms.get_all()
#     responses.append(response_dict)
#     dalybms.disconnect()

# # possibly average responses
```

request_motor_update

[4]:

```
import canopen

# network = canopen.Network()

# network.connect(channel='can0', bustype='socketcan')

# node = network.add_node(6, '/path/to/object_dictionary.eds')

# local_node = canopen.LocalNode(1, '/path/to/master_dictionary.eds')
# network.add_node(local_node)
```

request_temp_update

[ ]:

```
# TODO
```

request_system_update

[ ]:

```
# import read_battery_update
# import read_motor_update
# import read_temp_update
# import SQLite

# connect to database
# load and package system update
# insert into database
# disconnect
```

0.1 batch_send_updates

[ ]:

```
# import SQLite

# connect to local SQLite database
# connect to remote MySQL database
# query local database
# package batch update
# send batch to remote MySQL database
# wait for success
# if successful then purge all but most recent data from local SQLite database
# else ? # terminate? set a max on local database and lose data until cell␣
 ↪reception returns
```