General Flow:

1. Call to init() - Open files for read and write:
   - /*dev*/rfid_rc522_dev – A char dev from rfid module.
   - /dev/lcdi2c – A char dev from lcd module.
2. Application prints main menu to screen.
3. User choose which action to perform:

   - **Read card serial number –** will invoke `MFRC522_ReadSerialNum()` that sends `ioctl command` *`GET_ID`* (with userspace buffer 'a') to the rfid kernel module through the char dev. Then the driver will fill the buffer with the serial number of the card.

   - **Write user name –** will invoke `MFRC522_WriteUserNameToCard()` that will send `iotcl command` *`CHANGE_BLOCK`* to tell rfid driver which memory block number(1) of the card we want to write to. Next step is to receive from user the user name, and after that we write the user name to the card by a write command to char dev.
     Last step is to display the username that we wrote to the card on the Lcd by writing it to lcdi2c char dev, then send ioctl command *`BEEP`* to rfid module to generate a sound from buzzer.

   - **Write user id –** will invoke `MFRC522_WriteUidToCard()` that will send iotcl `command` *`CHANGE_BLOCK`* to tell rfid driver which memory block number(2) of the card we want to write to. Next step is to receive from user the user id, and after that we write the user id to the card by a write command to char dev.
     Last step is to display the user id that we wrote to the card on the Lcd by writing it to lcdi2c char dev, then send ioctl command *`BEEP`* to rfid module to generate a sound from buzzer.

   - **Read user name –** will invoke `MFRC522_ReadUserNameFromCard()`, call to `readBlockFromCard()` that will send ioctl *`CHANGE_BLOCK`* to change the driver to read from the desired memory block number (1 where username is stored) of the card, then read from rfid by reading from the char dev and storing the results in a memory region that allocated earlier.
     That func will return a pointer to the memory region where the username is stored, after we got the user name it will be sent to the lcd by writing it to the char dev(lcdi2c), then will send ioctl *`BEEP`* to generate a sound from buzzer.

   - **Read user id –** will invoke `MFRC522_ReadUidFromCard()`, call to `readBlockFromCard()` that will send ioctl *`CHANGE_BLOCK`* to change the driver to read from the desired memory block number(2 where userid is stored) of the card, then read from rfid by reading from the char dev and storing the results in a memory region that allocated earlier.
     That func will return a pointer to the memory region where the user id is stored, after we got the user id it will be sent to the lcd by writing it to the char dev(lcdi2c), then will send ioctl *`BEEP`* to generate a sound from buzzer.

   - **Read Username & Userid –** will invoke `MFRC522_ReadUnameAndUid()`, call to `readBlockFromCard()` twice: one time with 1 as parameter to read user name and the second time with 2 as parameter to read user id from the card.
     After receiving the data, we merge between uname&uid and send it to the lcd by write to lcdi2c char dev and generate a buzzer sound by send ioctl *`BEEP`* to rfid char dev.
     In the end we send the data (username&uid) to a web server that contains a data base that stores username, uid and time stamp.

File:

lcdi2c.c | lcdi2c.h

- LCD Linux Module Driver files.
- Register char dev to communicate with userspace.
- Register i2c driver with Device-Tree to communicate with hardware through i2c bus.
- Provide API’s to read/write/configure the hardware.

rc522.c | rc522_api.c | rc522_api.h

- RF-ID RFC522 Linux Module Driver files.
- Register char dev to communicate with userspace.
- Register spi driver with Device-Tree to communicate with hardware through spi bus.
- Get a gpio descriptor from Device-Tree to control a buzzer.
- Provide API’s to read/write/configure the hardware.

rc522App.c | rc522App.h

- Linux Userspace Application files.
- Print actions menu to the user.
- Write user name & user id to RFC522 cards.
- Read user name & user id from RFC522 cards.
- Display name & uid on the LCD module.
- Store the information from card in a database.