```
/*******************************************************************************
 Module
   FarmerMasterSM.c

 Revision
   1.0.1

 Description
   The receiving state machine for the Farmer

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 05/20/17 1:51    bag               created for the project
*******************************************************************************/
/*----------------------------- Include Files -----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine
*/

#include "ES_Configure.h"
#include "ES_Framework.h"
#include "FarmerMasterSM.h"
#include "FarmerTXSM.h"
#include "FarmerRXSM.h"
#include "Constants.h"
#include "LEDBlinkSM.h"
#include "Hardware.h"

#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_nvic.h"
#include "inc/hw_uart.h"
#include "inc/hw_sysctl.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"      // Define PART_TM4C123GH6PM in project
#include "driverlib/gpio.h"
#include "driverlib/uart.h"

/*----------------------------- Module Functions ---------------------------*/
/* prototypes for private functions for this machine.They should be functions
   relevant to the behavior of this state machine
*/
static void ProcessPairAck(void);
static void ProcessEncrReset(void);
static void ProcessStatus(void);
static void LED_Setter(void);
/*---------------------------- Module Variables ---------------------------*/
// everybody needs a state variable, you may need others as well.
// type of state variable should match htat of enum in header file

static FarmerMasterState_t CurrentState;
static uint8_t MyPriority;
static uint8_t DogSelect;

#define DOGTAG 101


/*------------------------------ Module Code ------------------------------*/
/*******************************************************************************
 Function
```

```
     InitFarmerMasterSM

 Parameters
     uint8_t : the priorty of this service

 Returns
     bool, false if error in initialization, true otherwise

 Description
     Saves away the priority, sets up the initial transition and does any
     other required initialization for this state machine
 Notes

 Author
     Matthew W Miller, 5/13/2017, 17:31
****************************************************************************/
bool InitFarmerMasterSM(uint8_t Priority)
{
      // state is unpaired
      CurrentState = Unpaired;
      // post entry event to self
      ES_Event EntryEvent;
      EntryEvent.EventType = ES_ENTRY;
      // set priority
      MyPriority = Priority;

      if (PostFarmerMasterSM(EntryEvent))
      {
            return true;
      }
      else
      {
            return false;
      }
}

/****************************************************************************
 Function
     PostFarmerMasterSM

 Parameters
     EF_Event ThisEvent , the event to post to the queue

 Returns
     boolean False if the Enqueue operation failed, True otherwise

 Description
     Posts an event to this state machine's queue
 Notes

 Author
     J. Edward Carryer, 10/23/11, 19:25
****************************************************************************/
bool PostFarmerMasterSM(ES_Event ThisEvent)
{
      // post event
      return ES_PostToService(MyPriority, ThisEvent);
}


/****************************************************************************
 Function
    RunFarmerMasterSM
```

```
 Parameters
   ES_Event : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   add your description here
 Notes
   uses nested switch/case to implement the machine.
 Author
   Matthew Miller, 05/13/17, 17:54
*****************************************************************************/
ES_Event RunFarmerMasterSM(ES_Event ThisEvent)
{
      ES_Event ReturnEvent;
      FarmerMasterState_t NextState;

      // set return event
      ReturnEvent.EventType = ES_NO_EVENT;

      // next state is current state
      NextState = CurrentState;

      // switch through states
      switch(CurrentState)
      {
            // if current state is unpaired
            case Unpaired:
                  // if event is entry
                  if(ThisEvent.EventType == ES_ENTRY)
                  {
                  //printf("FarmerMasterSM -- Unpaired -- ENTRY_EVENT\r\n");
                        // set the LED blink timer
                        // set blinker
                        // call LED function
                  }
                  // else if event is timeout
                  else if((ThisEvent.EventType == ES_TIMEOUT &&
ThisEvent.EventParam == CONN_TIMER))
                  {
                        // post entry event to self
                        ES_Event NewEvent;
                        NewEvent.EventType = ES_ENTRY;
                        PostFarmerMasterSM(NewEvent);
                  }
                  // else if event is right button down
                  else if(ThisEvent.EventType == ES_R_BUTTON_DOWN)
                  {
                        // increment the DOG selector
                        //TODO:This gives 0,1,2, but we want 1,2,3 FIX LATER
                        //printf("Dog Selection Button Pressed\r\n");
                        DogSelect = (DogSelect+1)%3;

                        //Tell the LED service to switch the LED to blink
                        ES_Event NewEvent;
                        NewEvent.EventType = ES_INCREMENT_LED;
                        PostLEDBlinkSM(NewEvent);
                  }
                  // else if the event is speech detected
                  else if(ThisEvent.EventType == ES_SPEECH_DETECTED)
                  {
                  //printf("FarmerMasterSM -- Unpaired -- SPEECH_DETECTED\r\n");
                        // set request pair in FARMER_TX_SM with DOG
```

```
                        setFarmerDataHeader(REQ_2_PAIR);
                        // set DogTag in FarmerTXSM


      /*****************************************************************/
                        //PICK THE DOG TO PAIR WITH
                        if(DogSelect == 0)
                        {
                              printf("FarmerMasterSM -- SENDING REQ2PAIR to DOG
1\r\n");
                              setDogTag(1);
                        }
                        else if(DogSelect == 1)
                        {
                              printf("FarmerMasterSM -- SENDING REQ2PAIR to DOG
2\r\n");
                              setDogTag(2);
                        }
                        else if(DogSelect == 2)
                        {
                              printf("FarmerMasterSM -- SENDING REQ2PAIR to DOG
3\r\n");
                              setDogTag(3);
                        }
                        else
                        {
                              printf("FarmerMasterSM --  UNRECOGNIZED
DOGTAG\r\n");
                        }
                        //setDogTag(DogSelect);
                        // Set destination address to BROADCAST since we are trying
to talk to everybody
                        setDestDogAddress(BROADCAST,BROADCAST); //TODO: replace
this with our xbee address so we dont piss off other teams

                        // next state is Wait2Pair
                        NextState = Wait2Pair;
                        // post entry event to self
                        ES_Event NewEvent;
                        NewEvent.EventType = ES_ENTRY;
                        PostFarmerMasterSM(NewEvent);

                        //Post ES_SEND_MESSAGE to FarmerTX
                        NewEvent.EventType = ES_SEND_RESPONSE;
                        PostFarmerTXSM(NewEvent);

                        //start 1s connection timer
                        ES_Timer_InitTimer(CONN_TIMER, CONNECTION_TIME);

                  }
                  break;
            // else if current state is Wait2Pair
            case Wait2Pair:
                  // if event is entry
                  if(ThisEvent.EventType == ES_ENTRY)
                  {
                        //printf("FarmerMasterSM -- Wait2Pair -- ENTRY\r\n");
                        // set the LED blink timer
                        // toggle the Blink LED
                  }
                  // else if event is timeout
                  else if(ThisEvent.EventType == ES_TIMEOUT &&
ThisEvent.EventParam == LED_TIMER)
                  {
```

```
                        // post entry event to self
                        ES_Event NewEvent;
                        NewEvent.EventType = ES_ENTRY;
                        PostFarmerMasterSM(NewEvent);
                    }
                    // else if event is Lost connection
                    else if(ThisEvent.EventType == ES_LOST_CONNECTION ||
(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam == CONN_TIMER))
                    {
                        //printf("FarmerMasterSM -- Wait2Pair --
LOST_CONNECTION\r\n");
                        // next state is Unpaired
                        NextState = Unpaired;
                        // post entry event to self
                        ES_Event NewEvent;
                        NewEvent.EventType = ES_ENTRY;
                        PostFarmerMasterSM(NewEvent);

                        //let the FarmerRXSM know we have lost connection
                        NewEvent.EventType = ES_LOST_CONNECTION;
                        PostFarmerRXSM(NewEvent);
                    }
                    // else if we receive a PAIR_ACK
                    else if((ThisEvent.EventType == ES_MESSAGE_REC) && (getHeader()
== PAIR_ACK))
                    {
                        //printf("FarmerMasterSM -- Wait2Pair -- PAIR_ACK
RECEIVEDL\r\n");
                        //printf("FarmerMasterSM -- Wait2Pair -- MESSAGE RECEIVED
-- HEADER = %i \r\n",getHeader());

                        ProcessPairAck();
                        //printf("FarmerMasterSM -- Wait2Pair --
SENDING_ENCRYPTION\r\n");

                        //Post ES_SEND_RESPONSE to FarmerTXSM
                        ES_Event NewEvent;
                        NewEvent.EventType = ES_SEND_RESPONSE;
                        PostFarmerTXSM(NewEvent);

                        /*
                        // Next state is Wait2Encrypt
                        NextState = Wait2Encrypt;
                        //printf("FarmerMasterSM -- Wait2Pair -- MOVING TO
Wait2Encrypt\r\n");
                        */

                        NextState = Paired;


                        //Set the LED solid
                        NewEvent.EventType = ES_PAIR_SUCCESSFUL;
                        PostLEDBlinkSM(NewEvent);

                        //turn on sound
                        HWREG(GPIO_PORTD_BASE + (ALL_BITS + GPIO_O_DATA)) |=
(SPEAKER_PIN_D);

                        //start 300ms message timer
                        ES_Timer_InitTimer(TRANS_TIMER, TRANSMISSION_RATE);

                        //restart 1s connection timer
                        ES_Timer_InitTimer(CONN_TIMER, CONNECTION_TIME);
```

```
                }
                break;


            // else if state is paired
            case Paired:

                if((ThisEvent.EventType == ES_MESSAGE_REC) && (getHeader() ==
STATUS) && (getDogAddrMSB() == getDestAddrMSB()) && (getDogAddrLSB() ==
getDestAddrLSB()))
                {
                      //handle the status message
                      //printf("FarmerMasterSM -- Paired -- Status
Received\r\n");
                      ProcessStatus();

                      //restart the 1s connection timer
                      ES_Timer_InitTimer(CONN_TIMER, CONNECTION_TIME);
                }

                else if((ThisEvent.EventType == ES_MESSAGE_REC) && (getHeader()
== ENCR_RESET) && (getDogAddrMSB() == getDestAddrMSB()) && (getDogAddrLSB() ==
getDestAddrLSB()))
                {
                      //handle the status message
                      //printf("FarmerMasterSM -- Paired -- Encr Reset
Received\r\n");
                      ProcessEncrReset();

                      //restart the 1s connection timer
                      ES_Timer_InitTimer(CONN_TIMER, CONNECTION_TIME);
                }

                //if the transmit timer times out
                else if((ThisEvent.EventType == ES_TIMEOUT) &&
(ThisEvent.EventParam == TRANS_TIMER))
                {
                      //header should already be set to a CTRL message I think
                      setFarmerDataHeader(CTRL);

                      //Post a send message event to FarmerTXSM
                      ES_Event NewEvent;
                      NewEvent.EventType = ES_SEND_RESPONSE;
                      PostFarmerTXSM(NewEvent);

                      //Restart 300ms message timer
                      ES_Timer_InitTimer(TRANS_TIMER, TRANSMISSION_RATE);

                }

                // if event is right button down
                else if(ThisEvent.EventType == ES_R_BUTTON_DOWN)
                {
                      // set right brake active in TX
                      //printf("Right Brake Engaged\r\n");
                      EnableRightBrake();
                }
                // else if event is right button up
                else if(ThisEvent.EventType == ES_R_BUTTON_UP)
                {
                      // set right brake inactive in TX
                      //printf("Right Brake Disengaged\r\n");
                      DisableRightBrake();
                }
```

```
                    // else if event is left button down
                    else if(ThisEvent.EventType == ES_L_BUTTON_DOWN)
                    {
                          // set left brake active in TX
                          //printf("Left Brake Engaged\r\n");
                          EnableLeftBrake();
                    }
                    // else if event is left button up
                    else if (ThisEvent.EventType == ES_L_BUTTON_UP)
                    {
                          // set left brake inactive in TX
                          DisableLeftBrake();
                          //printf("Left Brake Disengaged\r\n");
                    }
                    // else if event is reverse button down
                    else if(ThisEvent.EventType == ES_REV_BUTTON_DOWN)
                    {
                          // set reverse active in TX
                          //printf("Reverse Button Engaged\r\n");
                          EnableReverse();
                    }
                    // else if event is reverse button up
                    else if(ThisEvent.EventType == ES_REV_BUTTON_UP)
                    {
                          // set forward active in TX
                          //printf("Reverse Button Disengaged\r\n");
                          DisableReverse();
                    }
                    // else if event is peripheral button down
                    else if(ThisEvent.EventType == ES_P_BUTTON_DOWN)
                    {
                          // toggle peripheral in tx
                          //printf("Peripheral Button Engaged\r\n");
                          TogglePeripheral();
                    }
                    // else if event is lost connection
                    else if((ThisEvent.EventType == ES_LOST_CONNECTION) ||
((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam == CONN_TIMER)))
                    {
                          //printf("FarmerMasterSM -- Paired --
LOST_CONNECTIONr\n");

                          // post entry event to self
                          ES_Event NewEvent;
                          NewEvent.EventType = ES_ENTRY;
                          PostFarmerMasterSM(NewEvent);
                          // next state is unpaired
                          NextState = Unpaired;

                          //reset all of the control variables for the next pairing
                          clearControls();

                          //turn off vibration motors
                          SetDutyLeftVibrationMotor(0);
                          SetDutyRightVibrationMotor(0);

                          //let the FarmerRXSM know we have lost connection
                          NewEvent.EventType = ES_LOST_CONNECTION;
                          PostFarmerRXSM(NewEvent);

                          //let the blinker know we have lost connection
                          PostLEDBlinkSM(NewEvent);

                          //turn off sound
```

```
                        HWREG(GPIO_PORTD_BASE + (ALL_BITS + GPIO_O_DATA)) &=
~(SPEAKER_PIN_D);

                    }
                    break;
        } //end switch

        //Set current state to next state
        CurrentState = NextState;
        //return return event
        return ReturnEvent;

}

//Sequence of responses when we receive a PAIR_ACK
static void ProcessPairAck(void)
{
        //Set the data header to be an ENCR_KEY to prepare to send an encryption key
        setFarmerDataHeader(ENCR_KEY);
        setDestDogAddress(getDogAddrMSB(), getDogAddrLSB());
        //printf("FarmerMasterSM -- Process Pair Ack -- Dog to Pair with: %i   %i
\r\n", getDogAddrMSB(),getDogAddrLSB());
}

static void ProcessEncrReset(void)
{
        resetEncryptionIndex();
        //printf("FarmerMasterSM -- Process Encr Reset -- Encryption Index reset to:
%i  \r\n", getEncryptionKeyIndex());
}

static void ProcessStatus(void)
{
        setFarmerDataHeader(CTRL);

        //local variable GyroZ_MSB
        uint8_t GyroZ_MSB = getGyroZ_MSB();
        uint8_t vibration_duty;

        if(GyroZ_MSB >= 127)
        {
                vibration_duty = ((GyroZ_MSB-127)*100)/128;
                //printf("RIGHT VIBRATION MOTOR DUTY = %i\r\n", vibration_duty);
                SetDutyRightVibrationMotor(vibration_duty);
                SetDutyLeftVibrationMotor(0);
        }

        else if(GyroZ_MSB < 127)
        {
                vibration_duty = ((126-GyroZ_MSB)*100)/126;
                //printf("LEFT VIBRATION MOTOR DUTY = %i\r\n", vibration_duty);
                SetDutyLeftVibrationMotor(vibration_duty);
                SetDutyRightVibrationMotor(0);
        }

}

/*****************************************************************************
 private functions
 *****************************************************************************/


uint8_t getDogSelect(void)
{
```

```
        return DogSelect;
}
```